Table 7.4: TP rate (TPR) i TN rate (TNR) obtained by C4.5, SMO, IBk, XCS and UCS with the original domain and the re-sampled data sets.

	Original									
	C4.5		**SMO**		**IBk**		**XCS**		**UCS**	
	TPR	*TNR*	*TPR*	*TNR*	*TPR*	*TNR*	*TPR*	*TNR*	*TPR*	*TNR*
Original	98.40	97.31	0.00	100.00	95.20	98.85	85.60	96.15	100.00	100.00
Ovs	100.00	97.31	100.00	29.04	100.00	94.04	100.00	96.73	100.00	99.62
UnsTL	99.20	79.49	100.00	28.21	99.20	91.45	99.20	90.58	100.00	88.08
SMOTE	98.40	97.50	100.00	30.00	99.40	97.50	98.40	97.50	100.00	100.00
cSMOTE	96.22	97.27	100.00	29.43	97.84	98.54	95.20	98.27	92.80	99.62

be observed in other problems, where the application of a re-sampling technique induced the learner to misclassify regions that belonged to the majority class. In the case of SMO, there may be some intrinsic complexities of the domain that created especial difficulties to the system.

The results provided here visually illustrated the behavior of the different re-sampling techniques, giving a more detailed insight on how they work. Nonetheless, conclusions cannot be extracted from this simple case study. In the next section, we extend the analysis and compare the four re-sampling techniques in combination with the five learning algorithms on the large collection of imbalanced real-world problems used in the previous section.

7.5 Results on Re-sampled Domains

In this section, we analyze whether the application of the four re-sampling techniques presented above improves the performance of the five learning methods. As proceeds, we first introduce the experimental methodology, especially focusing on the steps taken to generate the re-sampled data sets. Then, we summarize the results obtained for each learning method and extract general conclusions about the excellence of each re-sampling technique. The full results are provided in appendix C.

7.5.1 Experimental Methodology

To compare the effect of the re-sampling techniques on each learning method, we employed the following methodology. We applied each re-sampling technique to each one of the training folds of the 25 data sets presented in section 7.3.1. This resulted in 100 new data sets, each one with 10 re-sampled training folds. The test folds were not modified so that the learners were tested with exactly the same information used in the original experiments (see section 7.3.2).

The five learning methodologies were configured as specified in section 7.3.1. No particular configuration was set for each re-sampling data set since we aimed at analyzing the impact of these re-sampling techniques to the system. The performance of each learner was measured with the product of TP rate and TN rate, since this metric is not biased toward the imbalance ratio of the learning data set. Also, the statistical procedure followed in section 7.3.1 was employed to compare the results. That is, the multiple-comparison non-parametric Friedman test (Friedman, 1937, 1940) was applied to contrast the null hypothesis that all the re-sampling techniques yielded the same results, on average, for a specific learner. If the multiple-comparison test rejected the null hypothesis, the post-hoc Nemenyi test (Nemenyi, 1963) was employed to detect significant differences among techniques.

With the purpose of compactness, the two next subsections present a summary of the results from which we extract general conclusions about the competitiveness of each re-sampling technique. We first provide the statistical analysis of the comparison of the four re-sampling techniques for each learner. The complete tables of results are supplied in appendix C. Thereafter, we summarize all the results in a table that gathers the average rank of each re-sampling technique for each of the five learning systems and highlight the main ideas and key conclusions from this summary.

7.5.2 Statistical Analysis of the Results

As proceeds, we statistically analyze the results of the comparisons of the four re-sampling techniques for each learner.

C4.5. The multiple-comparison Friedman test rejected the null hypothesis that the results obtained with the different re-sampling techniques were equivalent, on average, with $p = 0.0018$. Thus, we applied the post-hoc Nemenyi test, at $\alpha = 0.10$, to detect groups of re-sampling techniques which yielded equivalent results. Figure 7.7(a) connects the groups of learners that performed equivalently according to the Nemenyi test at $\alpha = 0.10$. Note that the test detected two groups. The first group comprised SMOTE, random over-sampling, and the original data set. The best ranked method was SMOTE.

The second group consisted of all the techniques except for SMOTE. Notice that the poorest results were obtained with the under-sampled data sets. This denoted that under-sampling the majority class might lead to sparsity problems in the particular data sets of the comparison since, on average, they contain a low number of instances. In general, the statistical analysis confirms the suitability of SMOTE and random over-sampling in combination with C4.5.

SMO. The multiple-comparison Friedman test rejected the hypothesis that all the re-sampling techniques resulted in the same performance, on average, with $p = 2.98 \cdot 10^{-6}$. Therefore, we applied the post-hoc Nemenyi test, whose results are provided in figure 7.7(b).

Several conclusions can be drawn from these results. Firstly, SMO and C4.5 benefited from different re-sampling techniques. In SMO, the best re-sampling technique was random over-sampling, which was also one of the best methods in C4.5. Nonetheless, notice that the second best ranked re-sampling method was under-sampling based on Tomek links, which, combined with C4.5, resulted in the poorest results. This highlights the idea that different learners benefit from different re-sampling methods.

Secondly, the statistical study detected that random over-sampling significantly outperformed the results obtained with the re-sampled data sets and the original data set. The Nemenyi test did not detect any further significant difference among the remaining re-sampling techniques and the original data set. Nevertheless, it is worth noting that the poorest average rank was obtained with the original data sets, which indicates that all re-sampling techniques are, on average, beneficial to SMO.

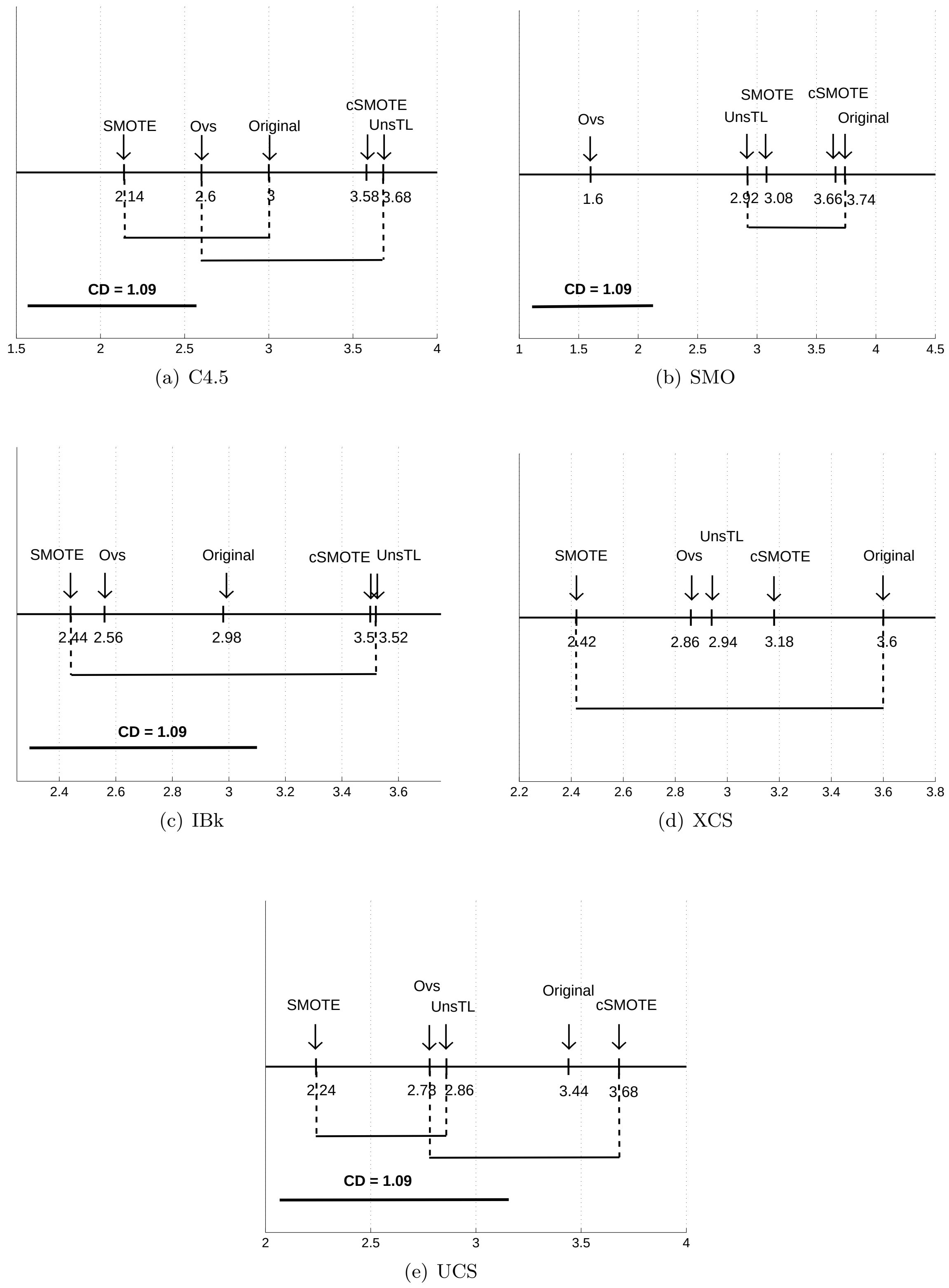


Figure 7.7: Comparison of the performance obtained by (a) C4.5, (b) SMO, (c) IBk, (d) XCS, and (e) UCS with the different re-sampling techniques. Groups of classifiers that are not significantly different at $\alpha = 0.10$ are connected.

IBk. The multiple-comparison Friedman test rejected the hypothesis that IBk obtained equivalent results with the original and re-sampled data sets with $p = 0.03$. However, the Nemenyi test, at $\alpha = 0.10$, could not find significant differences among the re-sampling methods. As follows, we provide some observations based on the average rank of each re-sampling algorithm, which is supplied in figure 7.7(c).

SMOTE was the best ranked method of the comparison, closely followed by random over-sampling. In the next positions of the ranking, we find the original data set, cSMOTE, and under-sampling based on Tomek links. Thence, under-sampling based on Tomek links provided, again, the poorest results. This was probably due to a problem of sparsity in the under-sampled data sets. Note that the position in the ranking of each re-sampling technique equals the corresponding position obtained by C4.5, although in C4.5 some of the differences were statistically significant.

XCS. The multiple-comparison Friedman test did not permit rejecting the hypothesis that the results obtained with the original data sets and the re-sampled data sets were equivalent, on average, at $\alpha = 0.05$ (the p-value returned by the test was $p = 0.1037$). Thence, as follows, we provide some remarks based on the average rank of each technique, which are illustrated in figure 7.7(d).

As observed for C4.5 and IBk, the two best ranked re-sampling techniques were SMOTE and random over-sampling. The third best method in the comparison was under-sampling with Tomek links. Thus, differently from IBk and C4.5—where under-sampling with Tomek links provided the worst performance—, the experimental results point out that XCS was not affected, on average, by decreasing the number of training examples, given that the points that lay close to the boundary were included in the final data set. Notice that the results achieved with the original data set were never significantly superior than those obtained with the under-sampled domain. A similar observation was drawn for SMO. This highlights that the competence of under-sampling with Tomek links was really dependant on the final learning system employed to learn the data model.

cSMOTE and the original data set come in the last positions of the ranking. Finally, note that the worst results were achieved with the original data sets. Therefore, as in SMO, all the re-sampling techniques appeared to improve the results obtained by XCS with the original data sets.

UCS. The multiple-comparison Friedman test rejected the null hypothesis that all the models extracted with the different re-sampled data sets and the original data set were equivalent, on average, with $p = 0.01$. Figure 7.7(e) groups the learners that performed equivalently according to the Nemenyi test at $\alpha = 0.10$.

The statistical analysis identified the same three best ranked methods as in XCS: SMOTE, random over-sampling, and under-sampling based in Tomek links. Therefore, the same conclusions provided in XCS can be extended to UCS. The poorest results were obtained with the original data sets and cSMOTE.

After analyzing each particular learner, the next section summarizes the results, gathering some key conclusions about the excellence of the different re-sampling techniques.

Table 7.5: Intra-method ranking for original and re-sampled data sets for C4.5, SMO, IBk, XCS, and UCS. Rows 1*st* to 5*th* indicate the number of times that each re-sampling technique was ranked in the correspondent position. The last column shows the average rank and its standard deviation.

	Resamp. Method	**1st**	**2nd**	**3rd**	**4th**	**5th**	**Avg. ± Std.**
C4.5	*original*	6	3	4	**9**	3	*3.00 ± 1.39*
	oversampling	7	4	**8**	4	2	*2.60 ± 1.26*
	undersampling TL	0	5	6	6	8	*3.68 ± 1.12*
	smote	**9.5**	**7.5**	4	3	1	***2.14 ± 1.17***
	csmote	2.5	5.5	2	5	**10**	*3.58 ± 1.44*
SMO	*original*	4	3	1.5	3.5	**13**	*3.74 ± 1.56*
	oversampling	**12**	**11**	2	0	0	***1.60 ± 0.63***
	undersampling TL	3	6.5	8	4.5	3	*2.92 ± 1.18*
	smote	3	4.5	**8.5**	5.5	3.5	*3.08 ± 1.20*
	csmote	2	1	4	**14.5**	3.5	*3.66 ± 1.03*
IBk	*original*	6	5	2.5	6.5	5	*2.98 ± 1.49*
	oversampling	3.5	**8.5**	**9.5**	2.5	1	*2.56 ± 0.98*
	undersampling TL	4	2	6	3	**10**	*3.52 ± 1.47*
	smote	**10.5**	3.5	2.5	6.5	2	***2.44 ± 1.44***
	csmote	1	5	5.5	**7.5**	6	*3.50 ± 1.17*
XCS	*original*	3	4	2.5	6	**9.5**	*3.60 ± 1.43*
	oversampling	7	**6**	3	1.5	7.5	*2.86 ± 1.61*
	undersampling TL	1	**6**	**11.5**	**6.5**	0	*2.94 ± 0.81*
	smote	**11**	4	1.5	5.5	3	***2.42 ± 1.51***
	csmote	3	4	7.5	**6.5**	4	*3.18 ± 1.23*
UCS	*original*	2	4	6	7	6	*3.44 ± 1.24*
	oversampling	**5.5**	**5.5**	6	5	3	*2.78 ± 1.32*
	undersampling TL	5	4	**8**	5.5	2.5	*2.86 ± 1.25*
	smote	7	11	3	2	2	***2.24 ± 1.18***
	csmote	**5.5**	0.5	1	**7.5**	**10.5**	*3.68 ± 1.55*

7.5.3 Summary

Having compared the performance obtained with each learner with the original and re-sampled data sets, the aim of this section is to summarize all the experimental analysis. For this purpose, table 7.5 supplies, for each learner, the number of occurrences of each re-sampling technique in each position of the ranking. For each classifier, the re-sampling method that is placed first in the ranking is marked in bold. The last column provides the average rank and the standard deviation for each re-sampling method, which are used to highlight the main observations and conclusions of the present comparison.

The results show that, in general, data set re-sampling yielded better learning performance than the one reached with the original data set. On average, the best results were achieved with *SMOTE* and *random over-sampling*. The empirical observations agree with some studies concluding that over-sampling is more effective than under-sampling in C4.5 (Japkowicz and

Stephen, 2002; Batista et al., 2004) and SMO (Japkowicz and Stephen, 2002). The results obtained herein allow us to extend this conclusion to IBk, XCS, and UCS. We hypothesize that under-sampling may cause a problem of sparsity as it removes instances that may be needed for learning. In fact, a deeper inspection of the detailed results (see appendix C) shows that under-sampling was better ranked in the problems *pim*, *wavd1*, *wavd2*, and *wavd3*, which have the highest number of instances per dimension[5], and poorly ranked in the problems with the lowest number of instances per dimension: *wdbc*, *wined1*, *wined2*, *wined3*, and *wpbc*.

The standard deviation of the rank somehow denotes the dependency of each re-sampling method on the characteristics of the training domain. For C4.5, SMOTE was the best ranked re-sampling method; besides, the rank deviation was small. In most of the cases, SMOTE was the first or the second method in the ranking. These results indicate that SMOTE should be used in combination with C4.5 to deal with class imbalances. For SMO, random over-sampling was the best ranked method and, at the same time, it showed a very low standard deviation. Consequently, SMO should be combined with random over-sampling in imbalanced domains. For IBk, SMOTE was the best ranked re-sampling technique followed very closely by random over-sampling. However, SMOTE had a much higher standard deviation, which indicates that its behavior highly depends on the domain. Therefore, this promotes the use of over-sampling in combination with IBk if we search for better robustness. For XCS, the best ranked re-sampling method, i.e., SMOTE, had one of the highest standard deviations. Thus, the behavior of this combination depended on the characteristics of the data. In this case, the practitioner may prefer to combine XCS with under-sampling based on Tomek Links, since its average rank was close to the SMOTE one and it had a very low standard deviation. For UCS, the best and the most robust re-sampling method was SMOTE.

Although the average results promote the application of re-sampling techniques to imbalanced domains, let us highlight that, in some cases, the best results were achieved with the original data set. That is, a detailed inspection of the results for each particular data set (see the full tables in appendix C) reveals that the performance of some learners on particular problems worsened when the data sets were re-sampled. This happened in problems such as *h-s*, *tao*, *wined1*, *wined2*, and *wined3*. In all these cases, some learners obtained a significantly lower performance than the one obtained with the original data set, which indicated that re-sampling was introducing some undesired characteristics to the training domain such as noise or new small disjuncts or niches. Having this in mind, the next section opens a discussion about several important aspects that have been made manifest throughout the comparative study, the answer of which will lead us to future work lines.

7.6 Discussion

The comparison performed in this chapter provided many valuable insights and enabled us to identify which learning systems were better than others and which combinations of learning algorithms with re-sampling techniques yielded the most accurate models on average. Nonetheless, we already pointed out a set of particular cases where the application of re-sampling techniques not only did not result in any improvement, but also provided significantly less accurate mod-

[5]The ratio between the number of instances and the number of attributes of a problem has been proposed elsewhere (Bernadó-Mansilla and Ho, 2005) as a measure of sparsity.

els. In general, after analyzing in detail all the results of the present comparison, the machine learning practitioner, who needs to solve a new problem, may still be wondering which learning algorithm and re-sampling technique he or she should use. In fact, these thoughts could be articulated in a more general way by posting the following two questions:

- *Why do some re-sampling techniques work well for some learners and data sets but fail with others?*
- *Why could not we find a re-sampling technique that works the best for all learners?*

These two questions are of especial interest in the current days, since the maturity of machine learning has resulted in the design of several algorithms to solve the same types of problems. Studies that mathematically analyze the properties of learning algorithms are not rare in the literature. However, they still cannot predict which learning algorithm will be the best performer for a new real-world problem, as long as the intrinsic characteristics of this problem are not known.

A particular contribution that provides a nice observation about these two questions can be found in the no-free-lunch (NFL) theorem (Wolpert, 1992, 1996). Loosely speaking, the NFL theorem mathematically demonstrates that, if we consider all the possible domains in our data set space, we can find as many data sets for which a learning algorithm 'A' outperforms another algorithm 'B' as viceversa. This theorem should be taken with a grain of salt, since we are not interested in all the possible domains, but in those domains that can be found in real-world problems. For example, we are not interested in domains with randomly generated instances, but in domains in which their instances define real-world concepts. Nevertheless, the NFL theorem gives mathematical formulation to a conclusion already observed in our experiments: that we cannot find a learning algorithm that performs the best for all possible domains, but for a particular range of them. Therefore, this conclusion demands the characterization of the real-world domains to relate them to the properties of each learning system, and so, to detect for which type of problems a learning algorithm is better suited than the others.

Some recent efforts have been taken to design measures to characterize classification problems. Michie et al. (1994) early highlighted the importance of domain characterization to analyze the performance of machine learning methods. The authors provided a set of measures—which consisted of statistical indicators and measures coming from the information theory—to characterize classification domains. These metrics were designed especially focusing on the analysis of decision trees. Later, Sohn (1999) used a more restricted set of metrics in a meta-model that compared the classification performance of several learning systems in terms of the data characterization. Although the results were promising, the authors already pointed out the need for further developing new metrics to capture more characteristics of the learning domains.

After these first promising works, Ho and Basu (2002) carefully examined the possible sources of data complexity and defined a set of measures that aimed at extracting some *geometrical characteristics* of the class distributions in the training data set. It was empirically shown that there was a considerable correlation between some of these measures and the error of some classifiers. For example, Bernadó-Mansilla and Ho (2005) and Bernadó-Mansilla et al. (2006) showed that the error of XCS was correlated with some metrics that estimated the length and linearity of the class boundary. The experimental results provided in these works evidenced that the characterization of the training data sets holds promise, being able to explain the behavior of learning algorithms on particular data, and so, starting to bridge the gap between artificial data sets with known characteristics—for which we have developed different models in chapters 5 and 6— and real-world data sets with unknown characteristics. Nonetheless, some drawbacks were also detected, such as the incapacity of the collection of designed metrics to explain all the sources of complexity of a classification problem.

As further work, this thesis proposes to continue with the study and design of new complexity metrics and apply them to

1. examine for which types of data a learning algorithm outperforms the others,
2. analyze the impact, from a geometrical point of view, of applying the re-sampling techniques to the different data sets, and
3. study the feasibility of applying each re-sampling technique with each particular learning system.

The aim of each one of these aspects is explained in what follows.

First, we will employ the complexity metrics to analyze which geometrical characteristics affect the different learners, identifying the sweet spot in which each learner is the best performer. Thence, we aim at answering questions such as *"Which learner for which real-world problem?"*. To achieve this, an API that includes all the complexity metrics proposed by Ho and Basu (2002), extends their definition enabling their application to data sets with continuous and nominal attributes and multiple classes, and provides new measures was implemented and made available as open source code (Orriols-Puig et al., 2008a). In addition, the correlation of different complexity metrics with the test error of XCS was analyzed (Bernadó-Mansilla et al., 2006). In further work, we aim at extending the analysis to other learners.

Second, the impact that each re-sampling technique has on the original data set—which has been visually illustrated for a particular case study in section 7.4.5—will be analyzed from the point of view of the change of the geometrical structure that the re-sampling techniques cause. That is, re-sampling techniques change the distribution of the training data set since new instances are added or some of the existing instances are removed from the original data set. Thus, in further work, it will be interesting to study how these re-sampling methods change the original distributions.

This change in the initial distribution may be either beneficial or detrimental for the learner employed in each particular case depending on the change of the initial distribution. For example, suppose that we over-sample a linearly-separable data set and that the resulting data is no longer linearly separable. In this case, the results obtained by a linear classifier in the re-sampled data set will be probably worse than those achieved with the original data. This is because there would be a misalignment between the re-sampling technique—and the changes that it produces to the original data—and the learning heuristic. Therefore, the aim of the third future work line is to identify which re-sampling technique is best suited for a particular learning algorithm given a real-world problem with a certain apparent complexity, providing answer—or, at least, some guidelines—to the question of *"Which machine learning technique combined with which re-sampling technique for which problem?"*.

7.7 Summary and Conclusions

In this chapter, we moved to real-world imbalanced classification problems and showed that both LCSs can successfully deal with the challenges posed by learning accurate models from rare classes in continuous domains with unknown characteristics. We designed a self-adaptation mechanism that uses a heuristic procedure to detect the maximum imbalance ratio between niches that lay closely in the solution space and adjust the parameters of both LCSs according to the theory presented in the last chapter. The excellence of the two LCSs was made evident through a comparison with three of the most influential machine learning techniques. XCS and UCS were the two best ranked methods in the comparison, providing statistically better results than the other methods in a considerably large number of data sets. Later, we introduced four re-sampling techniques in the comparison and analyzed the improvements that they supplied in combination with each one of the five learning techniques, extracting several observations for each particular case.

Two main conclusions, as well as several future line works, can be extracted from the overall analysis of this chapter. The first conclusion is that the two LCSs are two of the best options to extract classification models from imbalanced data sets, since they presented the best rank on average. Moreover, although it has not been further discussed here, LCSs have two important assets that differentiate them from the other three learning methods: (1) their online learning architecture and (2) their rule-based representation. The online learning architecture enables LCSs to learn from streams of data, which are very common in current scientific and industrial applications (Aggarwal, 2007; Gama and Gaber, 2007). The rule-based representation permits extracting models that can be read by human experts to some extend. This is similar to C4.5, which creates decision trees. Nonetheless, note the difference with respect to SMO and IBk. The models built by SMO consist of a set of support vector machines defined by weights, which can be barely interpreted. On the other hand, IBk does not create a general model.

The second conclusion is that, on average, re-sampling techniques improve the discovery of rare classes. In particular, SMOTE and random over-sampling appeared as the two best re-sampling techniques. Therefore, according to the empirical evidence provided in the present comparison, the machine learning user may bet for a combination of LCSs and SMOTE or over-sampling to deal with new challenging imbalanced problems. Nevertheless, we have also discussed that the intrinsic complexities of each particular problem may need different treatments. This leads us to consider the study of domain characterization as further work.

Chapter 8

Fuzzy-UCS: Evolving Fuzzy Rule Sets for Supervised Learning

In the last three chapters, we studied the capabilities of XCS and UCS to learn from imbalanced domains, addressing the first challenge proposed in this thesis. Along the study, as we were concerned about modeling rare classes, we evaluated the quality of the learners with metrics related to the accuracy per class. Nevertheless, in addition to model's accuracy, human experts may require the creation of legible models so that they can understand why a particular machine learning technique returns an output for a given unlabeled example. In some domains, such in medical diagnosis, human experts are sometimes more interested in the explanation that yields a prediction than in the prediction itself (Robnik-Sikonja et al., 2003). Although both LCSs use a rule-based representation, the interpretability of their models can be impaired by (1) the large number of rules that they tend to create (Bernadó-Mansilla and Ho, 2005; Bacardit and Butz, 2004; Wilson, 2002a; Dixon et al., 2004; Fu et al., 2001) and (2) the use of reasoning mechanisms that may be counter-intuitive to human experts. This is not a particular problem of LCSs, but it is shared by other machine learning techniques. In order to solve it, several authors have proposed the use of fuzzy logic (Zadeh, 1965, 1973) to create machines that can extract legible models and that use reasoning mechanisms closer to human ones.

The purpose of this chapter is to incorporate fuzzy logic concepts into LCSs with the aim of letting the systems evolve more legible classification models and use principled reasoning mechanisms defined in the fuzzy set theory. With this objective in mind, we take a creative process to mix the ideas that come from both fields and design Fuzzy-UCS (Orriols-Puig et al., 2007a,b, 2008b,e), the first Michigan-style LCS that evolves a fuzzy representation online for supervised learning tasks. The system is inspired by UCS, but it is completely redesigned to be able to deal with the new fuzzy representation. The competitiveness of the system is shown by comparing it with a large collection of fuzzy and non-fuzzy systems, which contains several of the most influential learners (Wu et al., 2007). In addition, we illustrate the added value of the online learning architecture of Fuzzy-UCS with respect to other learners by using Fuzzy-UCS to mine large volumes of data online.

The remainder of this chapter is organized as follows. Section 8.1 further discusses on the motivation of the present work. Section 8.2 introduces the basic concepts of fuzzy logics, presents some approaches in which GAs and fuzzy logics have been combined to create machine learning systems, and reviews some particular LCSs that use a fuzzy representation, highlighting the key differences with respect to Fuzzy-UCS. Section 8.3 gives a detailed description of the proposed Fuzzy-UCS algorithm. Section 8.4 examines the sensitivity of Fuzzy-UCS to configuration parameters. Then, section 8.5 studies the limitations that a linguistic representation may impose and compares it to a more flexible fuzzy representation. Section 8.6 makes an extensive comparison of the Fuzzy-UCS representation with a set of fuzzy and general-purpose machine learning techniques, analyzing the differences between the learners in terms of test performance and interpretability of the models. Section 8.7 exploits the online architecture of Fuzzy-UCS to mine a large data set, the 1999 KDD Cup intrusion detection data set. Finally, section 8.8 gives a summary of the work and future lines of research and presents a SWOT analysis to reflect the strengths, the weaknesses, the opportunities, and the threats of the new system.

8.1 Why Using Fuzzy Logic in LCSs?

Pattern recognition (Theodoridis and Koutroumbas, 2006) is concerned with the design of algorithms that are able to extract novel, useful, and hidden patterns from repositories of data. In this context, a competent supervised learning technique is required to be able to (1) identify patterns hidden between a set of descriptive attributes and a dependent variable, i.e., the output or the class, (2) represent these patterns in some legible structure, and (3) generalize over the input patterns to produce compact representations. During the last few decades, a lot of research has been conducted to design approaches that totally or partially fulfill the aforementioned requirements (Mitchell, 1997, 2006). As proceeds, we discuss whether these three aspects are satisfied by Michigan-style LCSs and propose the use of fuzzy logic as a powerful mechanism to create highly legible models from domains with uncertainty and imprecision.

We have shown in the previous chapters that Michigan-style LCSs are one of the most competitive alternatives to generalize over the input patterns and extract highly accurate models from real-world data sets. Therefore, they fulfill the first and third requirement. Nonetheless, the second requirement is not completely achieved. That is, although most of the current implementations of Michigan-style LCS use a rule-based representation—and rules can be individually interpreted—, the readability of the whole rule set may be impaired since LCSs

1. tend to evolve models with a large number of overlapped semantic-free interval-based rules, and

2. use reasoning mechanisms that can be little intuitive to human experts.

As follows we further discuss these two arguments in the context of interval-based rule representation, since it is the most used representation to deal with continuous attributes in LCSs.

The first reason that may hamper the readability of the models evolved by Michigan-style LCSs is that these systems tend to evolve a large number of overlapped semantic-free interval-based rules to define complex class boundaries (Bernadó-Mansilla and Ho, 2005; Bacardit and Butz, 2004; Wilson, 2002a; Dixon et al., 2004; Fu et al., 2001). That is, XCS and UCS systems alike usually represent continuous attributes with semantic-free intervals. Here, we use the term semantic free to refer to the fact that the lower and upper limits of each interval of each rule are modified independent of the value of the same attribute in other rules or the value of other attributes. This has two negative effects on the readability of the final population. The first negative effect is that Michigan-style LCSs tend to evolve populations that contain rules which are highly overlapped. Particularly, the class boundary usually consists of a large number of overlapping rules that predict different classes (Bernadó-Mansilla and Ho, 2005). This is not only caused by the rule representation, but also by the online learning architecture; that is, as rules are evaluated online, similar rules are maintained in the population. This large number of rules may hamper the readability of the rule set. The second effect comes directly with the fact that attributes are defined by intervals, so defining abrupt boundaries of the region where the rule is applicable. That is, inside the covered region, the implication predicted by the rule is completely true. This is counterintuitive from a human point of view, seeming more reasonable to have rules in which the degree of matching decreases as the boundaries of the covered region are approached.

The second aspect that may have a negative influence in providing legible explanations to human experts is that the reasoning mechanisms used by LCSs to infer the class of test instances usually mix information of all the matching rules to predict the output class. For example, in the case of XCS, the reasoning mechanism is based on a weighted sum that involves the prediction, the fitness, and, indirectly, the numerosity—since it is included in the fitness—of each matching classifier. Therefore, this reasoning mechanism further impairs the interpretability of individual rules because the contribution of each individual rule to the final prediction is not clear.

These two problems are not only related to LCSs, but to many machine learning techniques. To improve models legibility, several authors have adhered to the use of fuzzy logic and fuzzy set theory (Zadeh, 1965, 1973) to define fuzzy systems, that is, systems that use fuzzy logic to create highly legible models that predict environments with uncertainty and imprecision. Basically, the fuzzy set theory provides a legible knowledge representation and a robust reasoning mechanism with a mathematical formulation. In the last few decades, fuzzy logic set theory has been applied to different machine learning techniques such as rule-based systems and GBML (Cordón et al., 2001a) or neural networks (Buckley and Hayashi, 1994, 1995).

The motivation of the work performed in this chapter is to design a new supervised machine learning technique which combines the ideas of accuracy-based LCSs, GAs, and fuzzy logic together. That is, the new approach will join the online evaluation capabilities of LCSs, the search robustness of GAs, and the legible representation and reasoning mechanisms of fuzzy systems. The new online system is addressed as Fuzzy-UCS. Therefore, Fuzzy-UCS aims at providing similarly accurate, but more readable models than those created by XCS and UCS by (i) using a more readable rule representation, since rule variables are represented by linguistic terms, and (ii) evolving smaller rule sets. It is worth noting that Fuzzy-UCS is not the first Michigan-style LCSs that uses a fuzzy representation, but it is the first *learning fuzzy-classifier system* (LFCSs) that works under a supervised learning scheme and builds the knowledge online.

Before proceeding with a detailed description of the learning architecture of Fuzzy-UCS, the next section provides a concise introduction to fuzzy logic, gives a general schema of how GAs have been used in fuzzy systems, and reviews some of the early approaches of LCSs that evolve a fuzzy representation, highlighting the differences with respect to the Fuzzy-UCS architecture.

8.2 Fuzzy Logics in GBML

Fuzzy Logic can be defined as an extension of the traditional logic systems, which has its origins in the ancient Greek philosophy. Fuzzy logics provides a conceptual framework that permits representing the knowledge in environments with uncertainty and imprecision, two characteristics that are really present in nature. In fact, the natural language itself abounds with vague and imprecise concepts. Therefore, fuzzy logic defines a set of procedures or modes of reasoning that are approximate rather than exact. In brief, it extends the concepts of "true" and "false" in bi-valued logic with a third value that indicates that something is "possible" and gives a numeric value between true and false. Recently, the notion of fuzzy logic gained importance due to the pioneer contributions of Zadeh (1965, 1973), who established the foundations of the *fuzzy set theory* and, by extension, of *fuzzy logic*. The following subsections briefly explain the basic mechanisms of fuzzy systems that will be required for the remainder of this chapter. Finally, the different formulas in which GAs and fuzzy logic have been used together in machine learning are briefly introduced, especially focusing on the existing proposals of LFCS.

8.2.1 Fuzzy Logic and Fuzzy Systems

Fuzzy Systems are fundamental methodologies to represent and process linguistic information. Fuzzy systems use fuzzy logic to either represent the knowledge or model the interactions and relationships among the system variables in environments where there is uncertainty and imprecision. Because of this ability to deal with ill-defined data, fuzzy systems have been widely applied to control, classification, and modeling problems (Klir and Yuan, 1995; Pedrycz, 1998) where classical tools were unsuccessful. In what follows, we briefly introduce the basic concepts of the fuzzy set theory and shortly review how they can be incorporated into rule-based systems.

In the fuzzy set theory, each fuzzy subset A of a "crisp"[1] set X is characterized by giving a *degree of membership* to each of its elements $x \in X$. Thence, given a certain observation x and a fuzzy set A, a function addressed as *fuzzy membership function* is defined to return a membership degree of x into A. For example, let us suppose that we define the fuzzy set that represents the term *old*. Then, given a new proposition x (e.g., $x =$"John is 54"), the fuzzy membership function would provide a degree of membership of x to the set A, which could be *absolutely true* (if John is old), *absolutely false* (if John is not old), or some *intermediate truth degree* (John is old with a degree of 0.75). Several propositions can be combined by connectives, e.g., conjunction, disjunction, and negation. Thence, the fuzzy set theory gives a mathematical interpretation to these connectives so that a new membership degree can be calculated from several propositions joined by connectives.

In particular, the fuzzy set theory has been successfully applied to rule-based systems, resulting in the so-called *fuzzy rule-based systems* (FRBSs). FRBSs consist of fuzzy rules, that is, *if-then* constructions that have the following general form:

$$\textbf{IF } x_1 \text{ is } A_1 \text{ and ... and } x_n \text{ is } A_n \textbf{ THEN } B, \tag{8.1}$$

where the variables of the antecedent and the consequent contain linguistic labels, that is, the labels are represented by fuzzy sets. These rules are usually called linguistic FRBS or Mamdani

[1]Crisp set is used to refer to a set that follows the bi-valued logic.

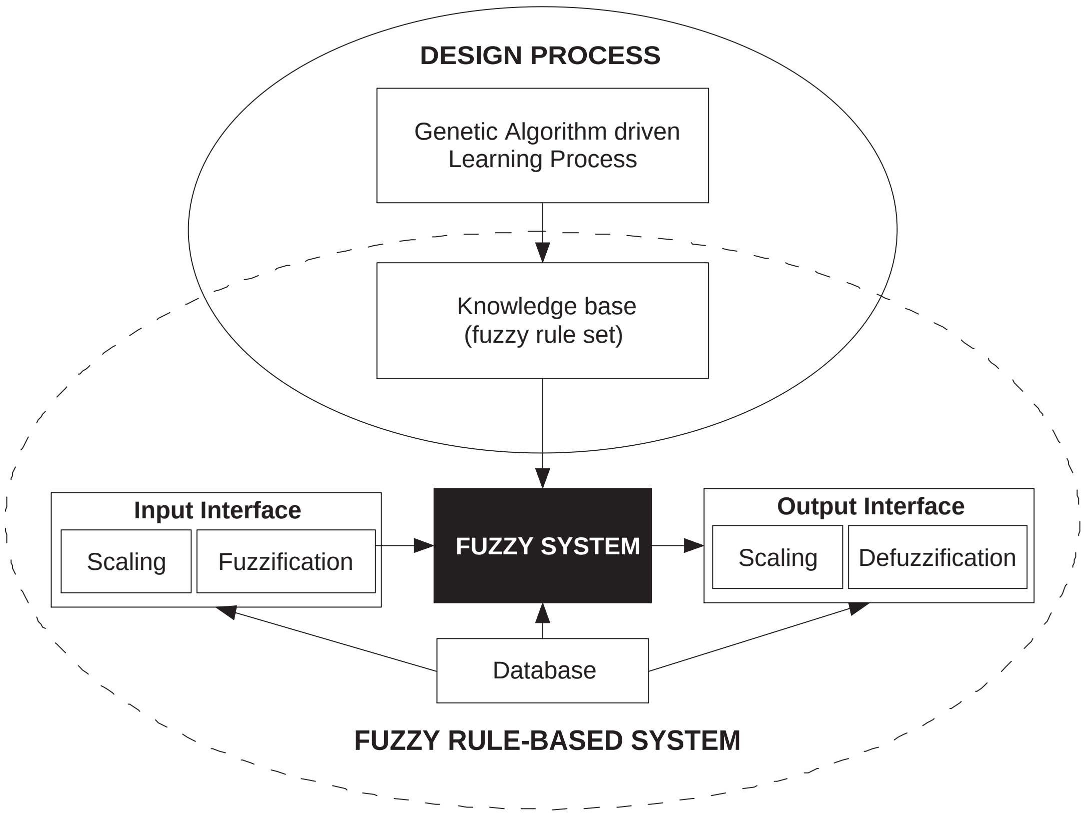


Figure 8.1: Schematic of GFRBS architecture.

FRBSs (Mamdani, 1974) and represent two essential advantages with respect to classical rule-based systems:

1. As variables use fuzzy sets, they naturally can handle uncertainty and vagueness.
2. Rule inference is driven by the reasoning methods defined in the fuzzy set theory, thus, providing inference with a mathematical framework.

Thereupon, the two main tasks in the development of a FRBS are: (1) generate a rule set that represents the problem domain accurately and (2) design an inference mechanism that permit combining the information of all the matching rules. In the next subsection we discuss different methodologies in which GAs are used to assist the building of FRBSs.

8.2.2 Genetic Algorithms in Fuzzy Systems

One of the main drawbacks associated with a FRBS is that the fuzzy rule set has to be defined by human experts, which may be a complex task in some real-world domains. In the last few decades, research has been conducted on designing methods to automatically extract fuzzy rules from a set of data, creating a model that represents the learning domain accurately. Many authors have regarded the rule extraction as an optimization problem where a set of rules, whose attributes are defined by linguistic terms, need to be found. Then, several search and optimization procedures can be applied to solve this problem.

One of the most prominent proposals is the use of GAs—and evolutionary algorithms in general—to generate this fuzzy rule set. At this point of the thesis, there is not need to further discuss the several reasons that promote the use of GAs, instead of other search mechanisms, to tune different components of FRBS. We have extensively discussed, and empirically shown in the previous sections, (1) the capability of GAs to deal with complex, large search spaces, (2) the success of the use of genetic search in machine learning, and (3) the flexibility of GAs that permit introducing a-priori knowledge. These three reasons are also applicable here, making GAs one of the most appealing methods to be combined with fuzzy systems.

The combination of GA with FRBS has lead to a new branch of GBML that has been addressed as *genetic fuzzy rule-based systems* (GFRBSs) (Cordón et al., 2001a). The basic idea of GFRBS is that GAs are used for either (i) tuning a pre-existing set of fuzzy rules typically with the aim of increasing the global accuracy of the system, as well as the readability of the fuzzy rule set, or (ii) generating the fuzzy rule set from scratch. To further illustrate this process, figure 8.1 provides a general schematic of a GFRBS, which is composed by a FRBS and a genetic procedure that guides the design process. The FRBS is formed by the *knowledge base*, which contains the fuzzy rule set, and an inference engine which, in turn, consists of:

1. An *input interface*, which transforms the input data into fuzzy sets by using a fuzzification process.
2. An *output interface*, which translates the fuzzy rule action to a real action by using a defuzzification method.
3. A *database*, which contains the definition of all the linguistic terms and the membership functions defining the semantics of the linguistic labels.

GAs have assisted the design of the FRBS in several ways. Following the taxonomy provided by Herrera (2008), GFRBS can be grouped in the following three classes.

- GFRBS in which the GA tunes some component of the rule set such as (i) the parameters of the membership functions of the linguistic terms used in the rule set (Casillas et al., 2005; Karr, 1991); (ii) the inference system itself (Alcalá-Fdez et al., 2007; Crockett et al., 2006, 2007); and (iii) the defuzzification function (Kim et al., 2002).
- GFRBS in which the GA is applied to learn some components of the GFRBS. Four approaches can be followed in this case, i.e., algorithms that (i) learn the knowledge base from a set of numerical data (Thrift, 1991); (ii) select the best rules extracted by another algorithm (Ishibuchi et al., 1995, 1997); (iii) learn first the database and then derive the best fuzzy rules for the selected database (Cordón et al., 2001b)—this process can be repeated to get the best combination of both database and knowledge base; or (iv) simultaneously learn both the database and the knowledge base (Homaifar and McCormick, 1995).
- GFRBS in which both the knowledge base and the inference engine parameters are tuned (Marquez et al., 2007).

From these three branches of GFRBS, we are concerned about the GFRBS that learn the rule set from scratch. More specifically, our aim is to design an online Michigan-style LCSs that learns fuzzy classification models with improved legibility. In the following subsection, we review few approaches that fall under this definition, emphasizing the key differences with respect to Fuzzy-UCS.

8.2.3 Related Work on Learning Fuzzy-Classifier Systems

Since Valenzuela-Rendón (1991) presented the first proposal of LFCSs, several authors have adhered to the idea of building machine learning techniques that evolve models online. Most of these systems were applied to solve reinforcement learning and control tasks. As follows, we present the most successful approaches to this topic and finally highlight the differences with respect to the underlying ideas of Fuzzy-UCS.

Valenzuela-Rendón (1991) introduced the first Michigan-style LFCS, which consisted of a fixed-size fuzzy-rule set and a fuzzy message list. The system was applied to solve function approximation tasks. The quality of the fuzzy rules was given according to the accuracy in which the output was estimated. Thus, the initial approach was not a pure reinforcement learning architecture. Later, Nomura et al. (1998) enhanced the system with true reinforcement learning.

Several strength-based Michigan-style LFCS have been proposed since this first description. Parodi and Bonelli (1993) presented an LFCS that automatically learned fuzzy relations, fuzzy membership functions, and fuzzy weights. The fitness (strength) of each rule was used for a double purpose. First, it served to compute the selection and replacement probability of the rule. Second, it permitted stronger rules to participate more soundly in the inference process.

Furuhashi et al. (1994) designed an LFCS that used multiple stimulus-response fuzzy rules operating in tandem. The system was applied to a control task in which a simulated ship had to reach a target without moving the obstacles found on its way. The same problem was addressed by Nakaoka et al. (1994) by using a single rule list. This approach avoided coverage problems in high dimensional spaces by using a dual fitness, one based on environmental payoff, and the other based on the accumulation of the level of activation during simulation.

Velasco (1998) defined a new LFCS architecture specifically designed for fuzzy process control. The system introduced the so-called limbos, i.e., a special workspace where new rules were generated and evaluated before being used in the real process plant. In this way, the system avoided using poorly-evaluated rules in the control system.

Ishibuchi et al. (1999b) designed one of the first proposals of LFCS for pattern classification. They used a fixed-size rule set where *don't care* symbols were defined to permit generalization in the fuzzy rules. A certainty factor, derived from a heuristic procedure prior to fitness evaluation, together with the predicted class formed the consequent of the rule. The rule consequent consisted of the class that the rule advocated and a certainty factor which was derived from a heuristic procedure prior to fitness evaluation. An evolutionary algorithm, which operated only on the rule antecedent, was responsible for creating promising new rules. Although the authors referred to the approach as a Michigan-style LFCS, the rule evaluation process was performed offline; that is, new rules were matched with all the examples of the training data set to compute their fitness. Therefore, this system was not able to deal with data streams. Recently, Ishibuchi et al. (2005) presented an hybridization of Pittsburgh-style and Michigan-style LFCSs. The system is mainly a Pittsburgh-style LCSs in which some individuals of the population can receive a local search procedure which is inspired by a Michigan-style LCSs. Again, the system evaluates the individuals offline.

Finally, the classic "competition versus cooperation" problem in genetic fuzzy systems was addressed by Bonarini (1996); Bonarini and Trianni (2001). Bonarini proposed a Michigan-style LCS called ELF, which faced the dilemma between the desired cooperation among fuzzy rules that match a given input state and the competition of these rules in the evolutionary algorithm. In ELF, the rule set was divided into several sub-populations, each one with the same antecedent. Then, the rules of different sub-populations cooperated to produce the control action, whilst the members of each subpopulation competed with each other. Moreover, ELF controlled the instability of general rules that participated in different sub-populations by providing each rule a reinforcement normalized on the difference between the maximum and the minimum reinforcement obtained by the subpopulation to which the rule belongs. In this way, ELF overcame some of the problems of strength-based LCSs. ELF was applied to several reinforcement learning problems, such as the coordination of autonomous agents.

All the LFCS described through this section are strength-based systems. In reinforcement learning, the first successful accuracy-based fuzzy rule-based system with generalization capability was proposed by (Casillas et al., 2007). To the best of our knowledge, no accuracy-based LFCS specifically designed for classification has been proposed until now. Therefore, Fuzzy-UCS is the first approach in this field. The system takes an accuracy-based approach to benefit from the advantages that these types of systems have introduced to LCSs, which are summarized as follows.

- Accuracy-based LCSs can distinguish over-general from accurate rules (Bull and Hurst, 2002).
- There are theoretical analyses that support the theory that, for binary representation, LCSs such as XCS will evolve a rule set with maximally-general and highly accurate rules if certain conditions are met (Butz et al., 2004b; Butz, 2006; Butz et al., 2007). Besides, there are further models, as those provided in chapters 5 and 6 that explain different facets of how these systems work. Although similar analyses in the continuous space are scarce, the positive conclusions extracted for the binary representation promote the use of Michigan-style LCSs.

The next subsection explains Fuzzy-UCS in detail.

8.3 Description of Fuzzy-UCS

The purpose of this section is to describe the design and implementation details of Fuzzy-UCS so that it can be used as an implementation guide. Figure 8.2 schematically illustrates the process organization of the system. The system works in two different modes: *exploration* or training and *exploitation* or test. In the exploration mode, Fuzzy-UCS seeks to evolve a maximally general rule set that minimizes the training error. In the exploitation mode, Fuzzy-UCS uses the evolved knowledge to infer the class of unlabeled examples. As proceeds, we first present the

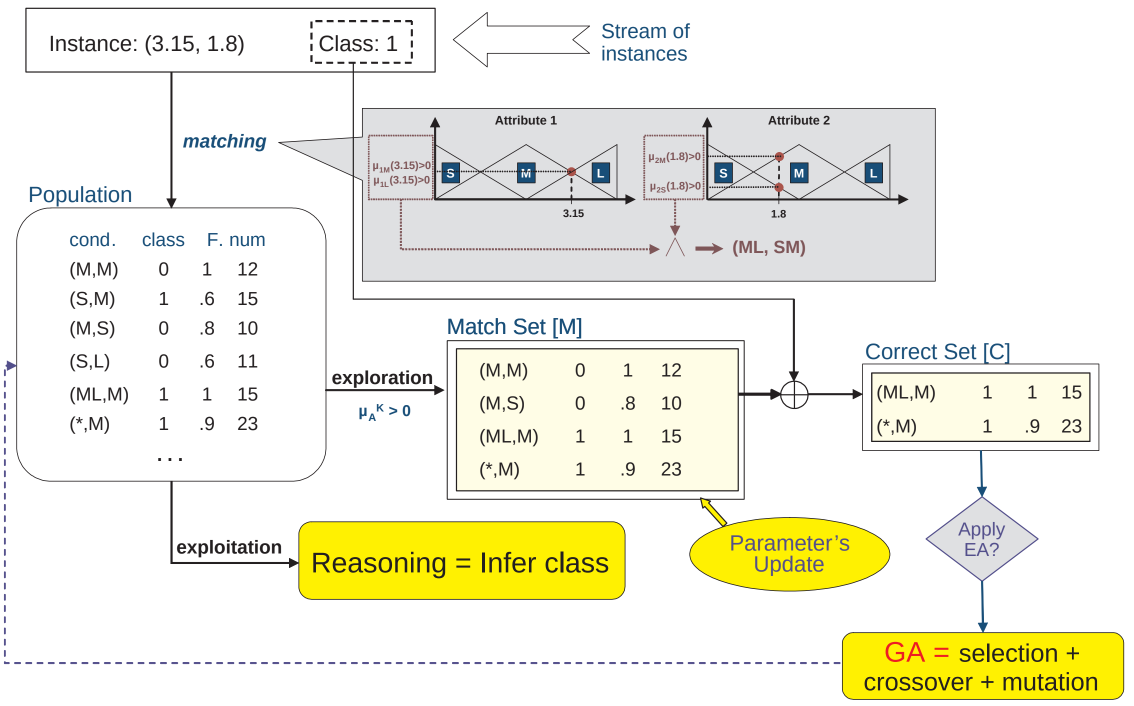


Figure 8.2: Schematic illustration of Fuzzy-UCS. The run cycle depends on whether the system is under exploration (training) or exploitation (test).

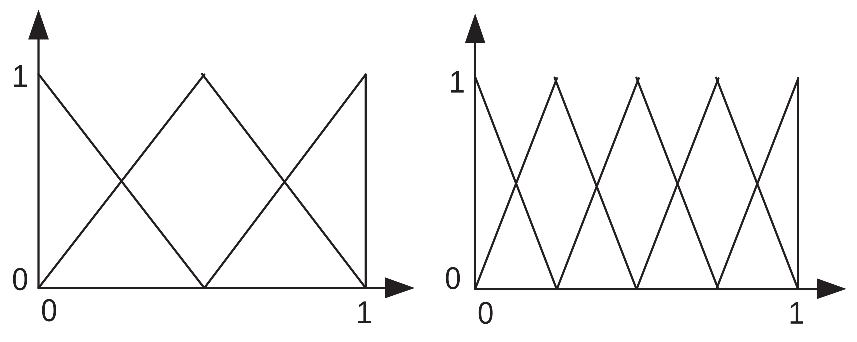
fuzzy knowledge representation used by Fuzzy-UCS and then introduce the learning interaction, the rule evaluation system, and the rule discovery component used in the training mode. Finally, we also introduce the reasoning mechanisms designed to infer the class of new unlabeled examples in test mode.

8.3.1 Knowledge Representation

We first introduce the fuzzy-rule-based representation of Fuzzy-UCS, explain how the matching mechanism works, and present the most relevant parameters that are associated with each classifier. Fuzzy-UCS evolves a *population* [P] of classifiers which jointly represent the solution to a problem. Each classifier consists of a rule whose condition is in *conjunctive normal form* and a set of parameters. The fuzzy rule follows the structure

$$\textbf{IF } x_1 \text{ is } \widetilde{A_1^k} \text{ and } \cdots \text{ and } x_n \text{ is } \widetilde{A_n^k} \textbf{ THEN } c^k \textbf{ WITH } w^k, \tag{8.2}$$

where each input variable x_i is represented by a disjunction of *linguistic terms* or *labels* $\widetilde{A_i^k} = \{ A_{i1} \vee \ldots \vee A_{in_i} \}$. In our experiments, all input variables share the same semantics, which are defined by means of triangular-shaped fuzzy membership functions. (see the examples of these semantics with different number of linguistic terms in figure 8.3). Note that this representation



(a) 3 linguistic labels (b) 5 linguistic labels

Figure 8.3: Representation of a fuzzy partition for a variable with (a) three and (b) five triangular-shaped membership functions.

intrinsically permits generalization since each variable can take an arbitrary number of linguistic terms. The consequent of the rule indicates the class c^k which the rule itself predicts. w^k is a weight ($0 \leq w^k \leq 1$) that denotes the soundness with which the rule predicts the class c^k. These types of rules with a weight in the consequent are known as fuzzy rules of type II (Cordón et al., 2001a).

The *matching degree* $\mu_{A^k}(e)$ of an example e with a classifier k is computed as follows. For each variable x_i, we compute the membership degree for each of its linguistic terms, and aggregate them by means of a T-conorm (disjunction). We enable the system to deal with missing values by considering that $\mu_{A^k}(e) = 1$ if the value e_i for the input variable x_i is not known. Then, the matching degree of the rule is determined by the T-norm (conjunction) of the matching degree of all the input variables. In our implementation, we used a *bounded sum* ($min\{1, a+b\}$) as T-conorm and the *product* ($a \cdot b$) as T-norm. Note that, if the fuzzy partition guarantees that the addition of all membership degrees is greater than or equal to 1—the membership functions used in our experiments satisfy this condition—, the selected T-norm and T-conorm allow for a maximum generalization. Therefore, an input variable x_i consisting of two consecutive linguistic terms will result in a matching degree of $\mu_{x_i}(e) = 1$ if the matching of e_i with both linguistic terms is greater than zero; thus, this choice supports the absence of the variable x_i.

Each classifier has four main parameters: 1) the fitness F, which estimates the accuracy of the rule; 2) the correct set size cs, which averages the sizes of the correct sets in which the classifier has participated (see section 8.3.2); 3) the experience exp, which computes the contributions of the rule to classify the input instances; and 4) the numerosity num, which counts the number of copies of the rule in the population. Some of the parameters such as the fitness and the experience have been "fuzzified" with respect the corresponding parameters in XCS and UCS.

To completely understand the new fuzzy rule representation, in the following subsections we detail how the different components of Fuzzy-UCS interact to evaluate the existing classifiers and create new promising rules.

8.3.2 Learning Interaction

Fuzzy-UCS inherits the process organization form UCS (see chapter 3), but it is adapted to deal with fuzzy rules. For this purpose, three main differences with respect to UCS need to be considered: the *matching calculation*, the *rule structure*, and the *inference methodology.*

1. *Matching calculation.* In UCS, the attributes are represented by intervals $[l_i, u_i]$, and thus, a rule matches an input example if $\forall e_i : l_i \leq e_i \leq u_i$. Therefore, the matching function returns a binary output indicating whether the classifier matches the example e or not. In Fuzzy-UCS, a rule k matches the input example with a matching degree $\mu_{A^k}(e)$, where $0 \leq \mu_{A^k}(e) \leq 1$. High values of $\mu_{A^k}(e)$ indicate that the prediction of rule k is fairly accurate.

2. *Rule structure.* In UCS, a rule predicts a single class with a certain fitness or quality. Consequently, the population may contain two rules with the same antecedent advocating different classes. To avoid this situation in Fuzzy-UCS, rules internally maintain a weight for each class that indicates the soundness in which this class is predicted. These weights are updated by the online learning architecture and are only used to determine the class that the rule predicts; that is, the class advocated by the rule is the class with the maximum weight. Therefore, the class predicted by the rule can change as the rule is evaluated online.

3. *Inference methodology.* In UCS, all the classifiers in [M] emit a fitness-weighted vote for the class they advocate, and the most voted class is chosen as the predicted output. In Fuzzy-UCS, different fuzzy-logic inference methods can be used to infer the class from the final fuzzy rule set (Cordón et al., 1999). Section 8.3.5 presents the three types of inference used by the system.

The learning organization of Fuzzy-UCS was redesigned considering these differences. As follows, the learning mechanism used during training is carefully reviewed, focusing on the main differences with respect to UCS. The reader is referred to section 8.3.5 for the details on the reasoning methods used to classify new instances in exploitation mode.

At each learning iteration, Fuzzy-UCS receives a new input example e and its class c, and the system builds the match set [M], which contains all the classifiers in [P] that have a matching degree $\mu_{A^k}(e)$ greater than zero.[2] Then, the system creates the correct set [C] with all the rules in [M] that advocate the class of the input example. If none of the classifiers in [C] match e with the *maximum matching degree*, the covering operator is triggered, which creates the classifier that maximally matches the input example. That is, for each attribute of the condition, we aggregate the linguistic term A_{ij} that maximizes the matching with the input value e_i. If e_i is not known, we randomly select a linguistic term and aggregate it to the attribute. Moreover, we introduce generalization by permitting the addition of other linguistic terms with probability $P_\#$. The initial values of the new classifiers are initialized according to the information provided by the current examples. Specifically, the fitness, the numerosity, and the experience are set to

[2]We do not require that rules have a matching degree greater than a certain threshold to be in [M], as sometimes done in regression (Casillas et al., 2007). In regression, the output is formed by means of aggregating rules with different actions. Thus, a minimum matching degree with the input may be required to participate in this process. However, in Fuzzy-UCS, the rules in [C] advocate the same class. In this way, Fuzzy-UCS avoids aggregating rules of different classes in the learning process, and so, a matching threshold appears to be less necessary.

1. The fitness of a new rule is set to 1 to give it opportunities to take over. Nonetheless, two important aspects should be noted. First, as the new classifiers participate in new match sets, their fitness and other parameters are quickly updated to their average values, and so, the initial value is not crucial. Second, as specified in the following sections, the system prevents young classifiers from having a strong presence in the genetic selection, and protects them from an early deletion. At the end of the covering process, the new classifier is inserted in the population, deleting another one if there is not room for it.

Next, in exploration mode, the classifiers in [M] that advocate the class c form the correct set [C]. As in UCS, the correct set works as a niche where the genetic algorithm is applied. Besides, after each learning iteration, the parameters of all the classifiers in [M] are updated. The following two subsections explicate these two procedures in detail.

8.3.3 Classifiers Update

At the end of each learning iteration, Fuzzy-UCS updates the parameters of the rules in [M]. As explained above, most of the parameters were redefined with respect to those of UCS to be able to deal with fuzzy rules—i.e, the parameters were "fuzzified". As proceeds, the equations used to update the parameters are provided.

First, the experience of the rule is incremented according to the current matching degree:

$$exp^k_{t+1} = exp^k_t + \mu_{A^k}(e). \tag{8.3}$$

Thence, in Fuzzy-UCS, the experience parameter accounts for the contributions of the classifier in matching instances; that is, classifiers that match with high degree several instances will have high experience. Next, the fitness is updated. For this purpose, each classifier internally maintains a vector of classes $\{c_1, \ldots, c_m\}$, each of them with an associated weight $\{v^k_1, \ldots, v^k_m\}$. Each weight v^k_j indicates the soundness with which rule k predicts class j for an example that fully matches this rule. These weights are incrementally updated during learning as explained as follows. The class c^k advocated by the rule is the class with the maximum weight v^k_j. Thus, given that the weights may change due to successive updates, the class that a rule predicts may also vary.

To update the weights, we first compute the sum of correct matchings cm^k for each class j:

$$cm^k_{j_{t+1}} = cm^k_{j_t} + m(k, j), \tag{8.4}$$

where

$$m(k,j) = \begin{cases} \mu_{A^k}(e) & \text{if j=c;} \\ 0 & \text{otherwise.} \end{cases} \tag{8.5}$$

Then, cm^k_{j+1} is used to compute the weights v^k_{j+1}:

$$\forall j : v^k_{j_{t+1}} = \frac{cm^k_{j_{t+1}}}{exp^k_{t+1}}. \tag{8.6}$$

For example, if a rule k only matches examples of class j, the weight v^k_j will be 1 and the remaining weights 0. Rules that match instances of both classes will have weights ranging from 0 to 1. Note that the sum of all the weights is 1.

The fitness is then computed from the weights with the aim of favoring classifiers that match examples of a single class. To carry this out, we use the following formula (Ishibuchi and Yamamoto, 2005):

$$F^k_{t+1} = v^k_{max_{t+1}} - \sum_{j|j \neq max} v^k_{j_{t+1}}, \tag{8.7}$$

where we subtract the values of the other weights from the weight with maximum value v^k_{max}. The fitness F^k is the value used as the weight w^k of the rule (see Equation 8.2). Note that this formula can result in classifiers with zero or negative fitness (for example, if the number of classes is greater than 2 and the class weights are equal). Lastly, the correct set size of all the classifiers in [C] is calculated as the arithmetic average of the sizes of all the correct sets in which the classifier has participated.

Finally, the rule k predicts the class c with the highest weight associated v^k_c. Thus, the class predicted is not fixed when the rule is created, and can change as the parameters of the rule are updated (especially during the first parameters updates). Once the parameters of all the classifiers in [M] have been updated, the GA can be applied to the current niche. In this case, the GA follows the process explained in the next subsection.

8.3.4 Classifiers Discovery

Fuzzy-UCS uses a steady-state niched *genetic algorithm* (GA) (Goldberg, 1989a) to discover new promising rules. The GA is applied to the classifiers that belong to [C]. Thus, the niching is intrinsically provided since the GA is applied to rules that match the same input with a degree greater than zero and advocate the same class.

The GA is triggered when the average time from its last application upon the classifiers in [C] exceeds the threshold θ_{GA}. It selects two parents p_1 and p_2 from [C] using proportionate selection (Goldberg, 1989a), where the probability of selecting a classifier k is

$$p^k_{sel} = \frac{(F^k)^\nu \cdot \mu_{A^k}(e)}{\sum_{i \in [C] | F^i \geq 0} (F^i)^\nu \cdot \mu_{A^k}(e)}, \tag{8.8}$$

where $\nu > 0$ is a constant that fixes the pressure toward maximally accurate rules (in our experiments, we set ν=10). Therefore, the probability of a classifier being selected depends on the product of its fitness and the matching degree with the input instance. Rules with negative fitness are not considered for selection. The two parents are copied into offspring ch_1 and ch_2, which undergo crossover and mutation with probabilities χ and μ respectively. The crossover operator crosses the antecedents of the rules by two points. The mutation operator checks whether each variable has to be mutated with probability μ. If so, three types of mutation can be applied: *expansion*, *contraction*, or *shift*. Expansion chooses a linguistic term not represented in the corresponding variable and adds it to this variable; thus, it can be applied only to variables that do not have all the linguistic terms. Contraction selects a linguistic term represented in the variable and removes it; so, it can be applied only to variables that have more than one linguistic term. By doing so, we avoid generating rules that do not match any example. Shift changes a linguistic term for its immediate inferior or superior.

The new offspring are introduced into the population. First, each classifier is checked for subsumption (Wilson, 1998) with their parents. Subsumption is a mechanism that prevents the creation of classifiers with specific conditions if there are more general and accurate classifiers in the population that cover the same region of the feature space. So, subsumption pressures toward maximally general and accurate classifiers. The process works as follows. If any parent's condition subsumes the condition of the offspring (i.e., the parent has, at least, the same linguistic terms per variable than the child), and this parent is highly accurate ($F^k > F_0^k$) and sufficiently experienced ($exp^k > \theta_{sub}$), the offspring is not inserted into the population and the numerosity of the parent is increased by one. Otherwise, we check [C] for the most general rule that can subsume the offspring. If no subsumer can be found, the classifier is inserted into the population.

If the population is full, excess classifiers are deleted from [P] with probability proportional to the correct set size estimate cs, following a method adapted from (Kovacs, 1999). Moreover, if the classifier is sufficiently experienced ($exp^k > \theta_{del}$) and the power of its fitness $(F^k)^\nu$ is significantly lower than the average fitness of the classifiers in [P] ($(F^k)^\nu < \delta F_{[P]}$, where $F_{[P]} = \frac{1}{N}\sum_{i\in[P]}(F^i)^\nu$), its deletion probability is further increased. That is, each classifier has a deletion probability p_k of:

$$p_k = \frac{d_k}{\sum_{\forall j \in [P]} d_j}, \tag{8.9}$$

where

$$d_k = \begin{cases} \frac{cs \cdot num \cdot F_{[P]}}{(F^k)^\nu} & \text{if } exp^k > \theta_{del} \text{ and } (F^k)^\nu < \delta F_{[P]}; \\ cs \cdot num & \text{otherwise.} \end{cases} \tag{8.10}$$

Thus, the deletion algorithm balances the classifier's allocation in the different correct sets by pushing toward deletion of rules belonging to large correct sets. At the same time, it favors the search toward highly fit classifiers, since the deletion probability of rules whose fitness is much smaller than the average fitness is increased.

8.3.5 Fuzzy-UCS in Test Mode

The aim of Fuzzy-UCS is to evolve a minimum set of maximally accurate rules that cooperate to cover all the input space. To achieve high classification accuracy, not only needs the system to create a population of highly accurate classifiers during learning, but it also has to define effective reasoning methods that use the information of the rule set to infer the class of new input examples. As these reasoning methodologies may not use all the rules in the inference process, rule set reduction techniques similar to those used in (Orriols-Puig and Bernadó-Mansilla, 2004) can be applied to remove the rules that are not considered for the reasoning technique. Herein, we discuss two different inference schemes. Furthermore, we present a reduction method for each one of these inference methods that permits a reduction the number of rules in the final rule set without decreasing training accuracy. Finally, we also introduce a third rule set reduction mechanism which allows for higher reductions, but does not guarantee that the reduced rule set results in the same training performance as the original one.

Class Inference

Once Fuzzy-UCS has evolved a population of highly general and accurate rules, this population is used to infer the class of new examples. Given a new unlabeled instance e, several rules predicting different classes can match (with different degrees) this instance. Thus, the knowledge contained in the set of matching classifiers has to be combined to decide the most likely output. For this purpose, several reasoning methodologies have been analyzed in the realm of fuzzy-rule based systems (Cordón et al., 1999; Ishibuchi et al., 1999a). Here, we adapt two inference approaches to Fuzzy-UCS. In both cases, only experimented rules ($exp^k > \theta_{exploit}$) are considered in the inference, where $\theta_{exploit}$ is a user-set parameter that indicates the minimum experience that a rule must have to participate in the inference process.

Weighted average inference. In this approach, all the experienced rules vote to infer the output. Each rule k emits a vote v_k for class j it advocates, where

$$v_k = F^k \cdot \mu_{A^k}(e). \tag{8.11}$$

The votes for each class j are added:

$$\forall j : vote_j = \sum_{k|c^k=j}^{N} v_k, \tag{8.12}$$

and the most-voted class is returned as the output.

Action winner inference. This approach selects the rule k that maximizes $\mu_{A^k}(e) \cdot F^k$, and chooses the class of the rule as output (Ishibuchi et al., 1999b). Thus, the knowledge of overlapping rules is not considered in this inference scheme.

Rule Set Reduction

At the end of the learning process, the population is reduced to obtain a minimum set of rules. We designed three types of reduction, which use one of the inference schemes presented above.

Reduction based on weighted average. Under the weighted average scheme, we reduce the final population by removing all the rules that a) are not experienced enough ($exp \leq \theta_{exploit}$) or b) have zero or negative fitness.

Reduction based on action winner. If action winner inference is used, only rules that maximize the prediction vote for a training example are necessary. Thus, after training, this reduction scheme infers the output for each training example. The rule that maximizes the vote v_j for each example is copied to the final population.

Reduction based on the fittest rules. This reduction tries to minimize the rule set size by selecting the most numerous and accurate rules for the final population. The methodology is a hybrid of the previous approaches. The reduction process is analogous to the reduction based on action winner, but now, the rule k that maximizes $F^k \cdot \mu_{A^k}(e) \cdot num^k$ for each input example is copied to the final population. By including the numerosity in the vote, we favor the most numerous and accurate rules. As this reduction may copy overlapping rules into the final population, weighted average is used to infer the class of a new example.

Overall, Fuzzy-UCS is an LFCS that evolves a set of linguistic fuzzy rules online and uses three different inference/reduction mechanisms to predict the class of test instances. Since Fuzzy-UCS is a brand new system that mixes different ideas coming from the LCSs, the GAs, and the fuzzy logic realms, the following sections take an empirical approach to analyze the behavior of the system. First, we study the influence of the different configuration parameters that have appeared along the description. Note that the majority of these parameters—or similar ones—are also present in XCS and UCS. Therefore, we hypothesize that they have a similar impact in the three systems; in any case, the next section empirically examines the effect of modifying the configuration parameters. This study results in a default configuration for Fuzzy-UCS, which is used in the remainder of this chapter. Subsequently, we analyze the differences between the three inference/reduction schemes of Fuzzy-UCS.

8.4 Sensitivity of Fuzzy-UCS to Configuration Parameters

In common with many competitive Michigan-style LCSs, Fuzzy-UCS has several configuration parameters, which enable to adjust the behavior of the system to evolve models of maximal quality for particular problems. At first glance, choosing a correct configuration may seem a crucial task only suitable to expert users. Nonetheless, several analyses identified the robustness of Michigan-style LCSs to the majority of configuration parameters. Actually, most of the applications of Michigan-style LCSs used the same default parameters to solve pattern recognition problems (Bernadó-Mansilla et al., 2002; Bernadó-Mansilla and Garrell, 2003; Butz, 2006; Orriols-Puig and Bernadó-Mansilla, 2008b; Dixon et al., 2002, 2004; Fu et al., 2001; Wilson, 2000). We consider that this robustness is also present in Fuzzy-UCS. Thence, this section empirically illustrates the behavior of Fuzzy-UCS with different configurations and relate this analysis to theoretical and empirical studies of the sensitivity of LCSs—particularly XCS and UCS—to configuration parameters. For the sake of compactness, here we also present the summary of the results and the statistical analysis that leads us to the most important conclusions. The current analysis is further detailed in appendix C.

Theoretical and empirical analyses of the sensitivity of LCSs[3] to configuration parameters detected four crucial parameters: (1) population initialization (Butz et al., 2001), (2) fitness pressure (Kharbat et al., 2005; Brown et al., 2007), (3) GA application rate (Butz et al., 2007), and (4) deletion pressure (Butz et al., 2007). The influence of the other parameters is less important, and most of LCSs works use a standard configuration for them.

Herein, we empirically study the sensitivity of Fuzzy-UCS to the configuration parameters. For this purpose, we analyzed the accuracy and size of the models evolved by Fuzzy-UCS related to the changes of four parameters or groups of parameters: (1) rules generalization in initialization, i.e., $P_{\#}$; (2) fitness pressure, i.e., ν; (3) setting of the genetic algorithm, i.e., θ_{GA}, θ_{del}, and θ_{sub}; and (4) deletion pressure, i.e., δ. We compared different configuration settings to the following default configuration (Cp), which sets the configuration parameters to standard values in literature: N=6 400, $F_0 = 0.99$, $\nu = 10$, $\{\theta_{GA}, \theta_{del}, \theta_{sub}\} = 50$, $\theta_{exploit} = 10$, $\chi = 0.8$, $\mu = 0.6$, δ=0.1, and $P_{\#} = 0.6$, Note that this configuration will be used in the following experiments of this chapter.

[3]These analyses refer to XCS and UCS, but could be easily extended to other Michigan-style LCSs.

Table 8.1: Properties of the data sets. The columns describe: the identifier of the data set (Id.), the name of the data set (dataset), the number of instances (#Ins), the total number of features (#Fea), the number of continuous features (#Cnt), the number of nominal features (#No), the number of classes (#C), the proportion of instances of the minority class (%Min), the proportion of instances of the majority class (%Maj), the proportion of instances with missing values (%MI), and the proportion of features with missing values (%MA).

Id.	**dataset**	**#Ins**	**#Fea**	**#Cnt**	**#No**	**#C**	**%Min**	**%Maj**	**%MI**	**%MA**
ann	Annealing	898	38	6	32	5	0.9	76.2	0	0
aut	Automobile	205	25	15	10	6	1.5	32.7	22.4	28
bal	Balance	625	4	4	0	3	7.8	46.1	0	0
bpa	Bupa	345	6	6	0	2	42	58	0	0
cmc	Contrac. method choice	1473	9	2	7	3	22.6	42.7	0	0
col	Horse colic	368	22	7	15	2	37	63	98.1	95.5
gls	Glass	214	9	9	0	6	4.2	35.5	0	0
h-c	Heart-c	303	13	6	7	2	45.5	54.5	2.3	15.4
h-s	Heart-s	270	13	13	0	2	44.4	56.6	0	0
irs	Iris	150	4	4	0	3	33.3	33.3	0	0
pim	Pima	768	8	8	0	2	34.9	65.1	0	0
son	Sonar	208	60	60	0	2	46.67	53.33	0	0
tao	Tao	1888	2	2	0	2	50	50	0	0
thy	Thyroid	215	5	5	0	3	14	60	0	0
veh	Vehicle	846	18	18	0	4	23.5	25.8	0	0
wbcd	Wisc. breast-cancer	699	9	9	0	2	34.5	65.5	2.3	11.1
wdbc	Wisc. diag. breast-cancer	569	30	30	0	2	37.3	62.7	0	0
wne	Wine	178	13	13	0	3	27	39.9	0	0
wpbc	Wisc. prog. breast-cancer	198	33	33	0	2	23.7	76.3	2	3
zoo	Zoo	101	17	1	16	7	4	40.6	0	0

Table 8.2: Configurations used to test the sensitivity of Fuzzy-UCS to configuration parameters.

	Cp	N=6 400, $F_0 = 0.99$, $\nu = 10$, $\{\theta_{GA}, \theta_{del}, \theta_{sub}\} = 50$, $\theta_{exploit} = 10$, $\chi = 0.8$, $\mu = 0.1$, δ=0.1, and $P_{\#} = 0.2$
$P_{\#}$	**C1**	$P_{\#} = 0.2$
	C2	$P_{\#} = 0.4$
ν	**C3**	$\nu = 1$
	C4	$\nu = 5$
θ_{GA},θ_{del},θ_{sub}	***C5***	$\theta_{GA} = \theta_{del} = \theta_{sub} = 100$ and $numIter = 100\,000$
	C6	$\theta_{GA} = \theta_{del} = \theta_{sub} = 200$ and $numIter = 100\,000$
	C7	$\theta_{GA} = \theta_{del} = \theta_{sub} = 100$ and $numIter = 200\,000$
	C8	$\theta_{GA} = \theta_{del} = \theta_{sub} = 200$ and $numIter = 400\,000$
δ	***C9***	$\delta = 1$

Table 8.3: Comparison of the sensitivity of Fuzzy-UCS to configuration parameters. Each cell shows the average rank of each configuration for a given inference scheme. The best ranked method is in bold. The $\ominus$ symbol indicates that the corresponding method significantly degrades the results obtained with the best ranked method.

		Performance			**Rule set size**		
		wavg	**awin**	**nfit**	**wavg**	**awin**	**nfit**
$P_\#$	**Cp**	1.83	1.83	**1.83**	**1.67**	**1.50**	1.75
	C1	2.42 $\ominus$	2.50 $\ominus$	2.25 $\ominus$	**1.75**	2.92 $\ominus$	3.00 $\ominus$
	C2	**1.75**	**1.67**	1.92	2.58 $\ominus$	1.58	**1.25**
ν	**Cp**	**1.25**	**1.42**	**1.42**	**1.08**	2.33 $\ominus$	2.67 $\ominus$
	C3	2.83 $\ominus$	2.75 $\ominus$	2.83 $\ominus$	3.00 $\ominus$	**1.25**	**1.17**
	C4	1.92	1.83	1.75	1.92 $\ominus$	2.42 $\ominus$	2.17 $\ominus$
θ_{GA}, θ_{del} & θ_{sub}	**Cp**	**1.92**	**2.13**	**2.13**	3.42 $\ominus$	3.17	3.17
	C5	4.00 $\ominus$	3.42	3.58 $\ominus$	3.42 $\ominus$	3.50	2.92
	C6	4.33 $\ominus$	4.63 $\ominus$	4.17 $\ominus$	**1.75**	2.83	**2.58**
	C7	2.33	2.25	2.29	3.42 $\ominus$	3.33	3.17
	C8	2.42	2.58	2.83	3.00	**2.17**	3.17
δ	**Cp**	**1.25**	1.54	**1.50**	**1.33**	1.75	1.75
	C9	1.75	**1.46**	**1.50**	1.67	**1.25**	**1.25**

We ran the experiments on a collection of real-world classification problems, whose characteristics are described in table 8.1. Due to the large number of tested configurations, we used a reduced collection of data sets to perform these experiments, that is: *bal*, *bpa*, *gls*, *h-s*, *irs*, *pim*, *tao*, *thy*, *veh*, *wbcd*, *wdbc*, and *wne*.

Table 8.2 summarizes the different configurations and the changes that they introduced with respect to the default configuration in each of the four experiments. Table 8.3 provides the average rank of the model's accuracy and size for each configuration and inference scheme. We divided the configuration settings into four groups, and each group was compared to the default configuration. The best ranked configurations for each comparison are marked in bold. The $\ominus$ symbol indicates that the corresponding configuration significantly degraded the results obtained with the best configuration according to a Bonferroni-Dunn test at $\alpha = 0.1$ (Dunn, 1961).

The results show that the generalization in the initial population is essential to the success of Fuzzy-UCS, supporting the theoretical analyses in the literature (Butz et al., 2001). For all the inference schemes, configurations Cp and $C2$ (i.e., $P_\# = \{0.6, 0.4\}$) were statistically equivalent, on average, and significantly better than $C1$ (i.e., $P_\# = 0.2$) in terms of accuracy. In terms of model size, the following significant differences were found: (i) for weighted average inference, Cp and $C1$ evolved the smallest rule sets; (ii) for action winner and fittest rules inference, $C1$ created significantly larger rule sets than Cp and $C2$. The last point can be easily explained as follows. As $C1$ used a low value of $P_\#$, the final populations contained more specific classifiers than populations created with Cp and $C2$. Action winner and fittest rules schemes only kept the classifiers that maximized the product of fitness and matching degree with a training instance in the final populations. As classifiers were more specific, a larger number of them were placed in the final population. On the other hand, with weighted average, the biggest population sizes were obtained with $C2$. This could be due to the existence of slightly general classifiers that were all maintained in the final population.

The second comparison shows the negative influence of having low fitness pressure. In terms of accuracy, better results were obtained as the fitness pressure increased (i.e., ν took higher values). Population sizes varied with the fitness pressure depending on the inference scheme. For weighted average inference, Cp led to the significantly smaller rule sets. This is because the fitness pressure drove toward a highly general and accurate set of rules. For the other two inference schemes, configuration $C1$ resulted in the significantly smaller rule sets. That is, as the fitness pressure was low, populations were full of over-general rules, which were kept in the final populations in detriment of fitter and more specific classifiers.

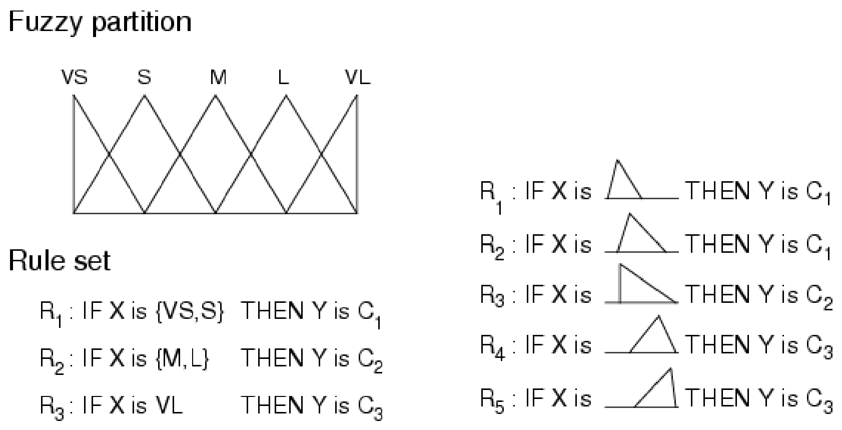
The third comparison shows the influence of the parameters related to the genetic algorithm, i.e., θ_{GA}, θ_{del}, and θ_{sub}. Initial intuition indicates that, if all niches receive the same number of genetic opportunities, the quality of the final models should remain the same. To test this, configurations $C7$ and $C8$ set $\theta_{GA} = \theta_{del} = \theta_{sub} = \{100, 200\}$ and increased $numIter = \{200\,000, 400\,000\}$ respectively. In this way, all niches received approximately the same number of genetic events. On the other hand, configurations $C5$ and $C6$ fixed $\theta_{GA} = \theta_{del} = \theta_{sub} = \{100, 200\}$ but maintained the same number of iterations as Cp. So, we expected that the quality of the models evolved by $C5$ and $C6$ was significantly lower than the quality of the models created by the three other configurations. This hypothesis was clearly supported by the experimental analysis, which showed that Cp, $C7$, and $C8$ resulted in the most accurate models. Moreover, significant differences on the population sizes were only found for the weighted average inference. The multiple-comparison test detected that the smaller models were created with configurations $C6$ and $C8$, the two configurations in which the period of application of the GA was higher.

Finally, the fourth comparison highlights the robustness of Fuzzy-UCS to the deletion pressure toward unfit classifiers, that is, the parameter δ. The pairwise analysis indicated that the hypothesis that configurations Cp and $C9$ are equivalent could not be rejected, according to a Wilcoxon signed-ranks test at $\alpha = 0.05$.

The study conducted in this section empirically showed that there are two crucial parameters to guarantee the success of Fuzzy-UCS: generalization in initialization $P_{\#}$ and fitness pressure ν. On the other hand, changing the setting of the other parameters had little effect on Fuzzy-UCS behavior. We acknowledge that better results could be individually obtained if we tuned Fuzzy-UCS for each particular problem. Nonetheless, as we are interested in robust systems that perform well on average, we use the default configuration for all the experiments in the following sections.

8.5 Knowledge Representation and Decision Boundaries

So far, we have described the Fuzzy-UCS classifier system with a *descriptive* or *linguistic* representation of fuzzy rules, which is referred to as linguistic Fuzzy-UCS in the remainder of this chapter, and have analyzed its robustness with respect to its configuration parameters. Linguistic rules are highly interpretable since they share common semantics; however, as this


Fuzzy partition
VS S M L VL
Rule set
R1 : IF X is {VS,S} THEN Y is C1
R2 : IF X is {M,L} THEN Y is C2
R3 : IF X is VL THEN Y is C3
R1 : IF X is THEN Y is C1
R2 : IF X is THEN Y is C1
R3 : IF X is THEN Y is C2
R4 : IF X is THEN Y is C3
R5 : IF X is THEN Y is C3

(a) Linguistic fuzzy rule set (b) Approximate fuzzy rule set

Figure 8.4: Graphical comparison between (a) linguistic and (b) approximate fuzzy rule sets.

representation implies the discretization of the feature space, a single rule may not have the required granularity to define the class boundary of a given domain accurately. Thus, Fuzzy-UCS would evolve a set of overlapping fuzzy-rules around the decision boundaries which match examples of different classes, and the output would depend on how the reasoning mechanism combines the knowledge of all these overlapping rules. Fuzzy-UCS includes three inference and reduction schemes which lead to a trade-off between the amount of information used for the inference process (i.e., the precision of the prediction) and the size of the rule set. Consequently, not only the linguistic representation but also the chosen inference and reduction schemes may impose a maximum limit on the accuracy rate that the system can reach.

To achieve better accuracy rates, several authors introduced the so-called *approximate* rule representation (also known as non-grid-oriented fuzzy systems, prototype-based representation, or fuzzy graphs) (Alcalá et al., 2001; Bardossy and Duckstein, 1995; Carse et al., 1996; Cordón and Herrera, 1997). This representation allows the variables of fuzzy rules to define their own fuzzy sets instead of representing linguistic variables. In this way, approximate fuzzy rules are semantic free; that is to say, the fuzzy sets of any variable of each rule can be independently tuned. However, this also results in a degradation of the interpretability of the final rule set, since the fuzzy variables no longer share a common linguistic interpretation. Figure 8.4 illustrates the two representations.

This section studies the interpretability-performance trade-off in Fuzzy-UCS and analyzes if the flexibility provided by the approximate representation allows the system to achieve higher levels of performance. For this purpose, we include the approximate representation in Fuzzy-UCS and adapt several mechanisms to deal with approximate rules. This alternative algorithm is described in the next section. This modification of Fuzzy-UCS is addressed as approximate Fuzzy-UCS.

Thus, our analysis consists of two parts:

- We first illustrate how both representations approximate the decision boundaries of an artificial problem with complex decision boundaries. The study demonstrates how the approximate representation can fit the training examples more accurately.

- Then, we compare the differences in terms of interpretability and accuracy between 1) the three inference schemes of linguistic Fuzzy-UCS and 2) linguistic Fuzzy-UCS versus approximate Fuzzy-UCS.

As follows, in section 8.5.1, we first design an approximate representation for Fuzzy-UCS. Then, sections 8.5.2 and 8.5.3 respectively develop each one of the two studies.

8.5.1 Approximate Fuzzy-UCS

In the approximate representation (Orriols-Puig et al., 2008g), the rule is similar to the descriptive one, but the variables in the rule condition take fuzzy sets instead of linguistic terms. Thus, the approximate fuzzy rule has the following form:

$$\textbf{IF } x_1 \text{ is } FS_1^k \text{ and } \cdots \text{ and } x_n \text{ is } FS_n^k \textbf{ THEN } c_j \textbf{ WITH } F^k, \tag{8.13}$$

where each variable x_i is represented by an independent fuzzy set FS_i, and each fuzzy set is defined by

$$FS_i = (a, b, c), \tag{8.14}$$

where a, b, and c are the x-axis value of the lower, middle and upper vertices of a triangular-shaped membership function, i.e,

$$\mu_{FS_i} = \begin{cases} \dfrac{x-a}{b-a}, & a \le x < b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & otherwise. \end{cases} \tag{8.15}$$

The operators that directly manipulate the rules were adapted to deal with the approximate representation. This includes matching, covering, crossover, mutation, and subsumption, which are explained in the following sections. Moreover, the inference process was also revised.

Matching. The matching operator calculates the matching degree of each input variable with its corresponding fuzzy set and aggregates all them by means of a T-norm (conjunction). As before, we used the product as T-norm. Note that the main difference with respect to linguistic Fuzzy-UCS is that, now, each variable is represented by a single semantic-free triangular shaped membership function.

Covering. The covering operator creates an independent triangular-shape fuzzy set for each input variable as follows.

$$a = rand\left(min_i - \frac{max_i - min_i}{2}, e_i\right); \tag{8.16}$$

$$b = e_i; \tag{8.17}$$

$$c = rand\left(e_i, max_i + \frac{max_i - min_i}{2}\right); \tag{8.18}$$

where min_i and max_i are the minimum and maximum value that the attribute i can take (both values are extracted from the training data set), e_i is the attribute i of the example e for which covering has been fired, and $rand$ generates a random number between both arguments. Thus, covering creates a triangle-shaped fuzzy set that maximally matches the input instance.

Crossover. The crossover operator generates a new offspring from two parents by crossing the rule antecedent as follows. First, is crosses the middle vertex b of the fuzzy membership function:

$$b_{child_1} = b_{parent_1} \cdot \alpha + b_{parent_2} \cdot (1 - \alpha); \tag{8.19}$$
$$b_{child_2} = b_{parent_1} \cdot (1 - \alpha) + b_{parent_2} \cdot \alpha; \tag{8.20}$$

where $0 \leq \alpha \leq 1$ is a configuration parameter. As we wanted to generate offspring whose middle vertex b was close to the middle vertex of one of his parents, we set $\alpha = 0.005$ in our experiments. Next, for both children, the procedure to cross the most-left and most-right vertices is the following. First, the two most-left and two most-right vertices are chosen

$$min_{left} = min(a_{parent_1}, a_{parent_2}, b_{child}); \tag{8.21}$$
$$mid_{left} = middle(a_{parent_1}, a_{parent_2}, b_{child}); \tag{8.22}$$
$$mid_{right} = middle(c_{parent_1}, c_{parent_2}, b_{child}); \tag{8.23}$$
$$max_{right} = max(c_{parent_1}, c_{parent_2}, b_{child}). \tag{8.24}$$

And then, these two values are used for generating the most-left and most-right vertices:

$$a_{child} = rand(min_{left}, mid_{left}); \tag{8.25}$$
$$c_{child} = rand(mid_{right}, max_{right}); \tag{8.26}$$

where the functions min, $middle$, and max return respectively the minimum, middle, and maximum values between their arguments.

Mutation. The mutation operator decides randomly if each vertex of a variable has to be mutated. The central vertex is mutated as follows:

$$b = rand(b - (b - a) \cdot m_0, b + (c - b) \cdot m_0), \tag{8.27}$$

where m_0 ($0 < m_0 \leq 1$) defines the strength of the mutation. The left-most vertex is mutated as

$$a = \begin{cases} rand\left(a - \frac{b-a}{2} \cdot m_0, a\right) & \text{if } F > F_0 \text{ \& no crossover} \\ rand\left(a - \frac{b-a}{2} \cdot m_0, a + \frac{b-a}{2} \cdot m_0\right) & \text{otherwise.} \end{cases} \tag{8.28}$$

And the right-most vertex

$$c = \begin{cases} rand\left(c - \frac{c-b}{2} \cdot m_0, c\right) & \text{if } F > F_0 \text{ \& no crossover} \\ rand\left(c - \frac{c-b}{2} \cdot m_0, c + \frac{c-b}{2} \cdot m_0\right) & \text{otherwise.} \end{cases} \tag{8.29}$$

That is to say, if the rule is accurate enough ($F > F_0$) and has not been generated through crossover, mutation forces to generalize it. Otherwise, it can be either generalized or specified. In this way, we increase the pressure toward maximally general and accurate rule sets.

Subsumption. Subsumption was redefined as follows. We considered that a classifier k_1, which is experienced enough ($exp^{k_1} > \theta_{sub}$) and accurate ($F^{k_1} > F_0$), could subsume another classifier k_2 if for each variable i

$$a^i_{k_1} \leq a^i_{k_2}; \tag{8.30}$$

$$c^i_{k_1} \geq c^i_{k_2}; \tag{8.31}$$

$$b^i_{k_1} - (b^i_{k_1} - a_{k^i_1}) \cdot \delta \leq b^i_{k_2} \leq b^i_{k_1} + (c^i_{k_1} - b^i_{k_1}) \cdot \delta; \tag{8.32}$$

where δ is a discount parameter (in our experiments we set $\delta = 0.001$). Thus, a rule's condition subsumes another if the supports of the subsumed rule are enclosed in the supports of the subsumer rule, and the middle vertex of their triangular-shaped fuzzy sets are close in the feature space.

Inference. Given a new test example, the most likely output is the class predicted by the rule k that maximizes $F^k \cdot \mu_{A^k}(e)$. We have considered this action winner scheme as inference process because the prototype-based representation considered in the approximate approach inherently advocates independence among the fuzzy rules.

8.5.2 Decision Boundaries: Study on an Artificial Domain

Before proceeding with a large comparison between the two types of representations and the three types of inference algorithms in the linguistic representation, we first analyzed linguistic Fuzzy-UCS and approximate Fuzzy-UCS on a case study. We also included UCS with interval-based representation (Bernadó-Mansilla and Garrell, 2003; Orriols-Puig and Bernadó-Mansilla, 2006b; Orriols-Puig and Bernadó-Mansilla, 2008a) in the analysis. We graphically studied how the two fuzzy representations approximated the decision boundaries of an artificially designed domain with respect to interval-based UCS. We chose a two-dimensional problem to facilitate the visualization: the *tao* problem (Bernadó-Mansilla et al., 2002) (see figure 8.5(a)). This problem presents curved-shaped boundaries, whose approximation poses a challenge to the linguistic fuzzy representation. Moreover, we compared the training accuracies, as well as the size of the evolved rule set. This analysis was restricted to the features of the tested problem, and only estimated the training error; thus, our aim was not to extract general conclusions, but to provide an intuitive visualization of the knowledge evolved by the different techniques. This analysis is complemented in the next section, where the three learners are compared in a set of real-world problems.

We configured UCS with the following parameter values: $numIter$=100 000, N=6 400, $acc_0 = 0.99$, ν=10, $\{\theta_{GA}, \theta_{del}, \theta_{sub}\}$=50, χ=0.8, μ=0.04, δ=0.1, r_0=0.2. For Fuzzy-UCS, we used the default configuration (see section 8.4), except for $P_\# = 0.2$. We modified this configuration parameter only for the case study; in all the remaining experiments, the default configuration is used. This change was because we aimed at initializing the population with quite specific rules since the problem has only two dimensions and a high density of instances. Besides, for the approximate representation we set $r_0 = 0.2$. The three types of inference presented in section 8.3.5 were used: weighted average (wavg), action winner (awin), and fittest rules (nfit). Figure 8.5(b) depicts the boundaries evolved by interval-based UCS. Figures 8.6, 8.7, and 8.8 report the decision boundaries for linguistic Fuzzy-UCS with weighted average inference, action winner inference, and fittest rules inference respectively. In each case, we experimented with

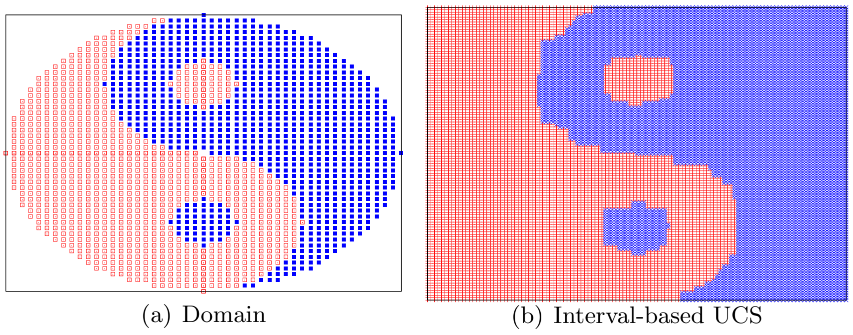

(a) Domain (b) Interval-based UCS

Figure 8.5: (a) Domain of the tao problem and (b) decision boundaries obtained by UCS.

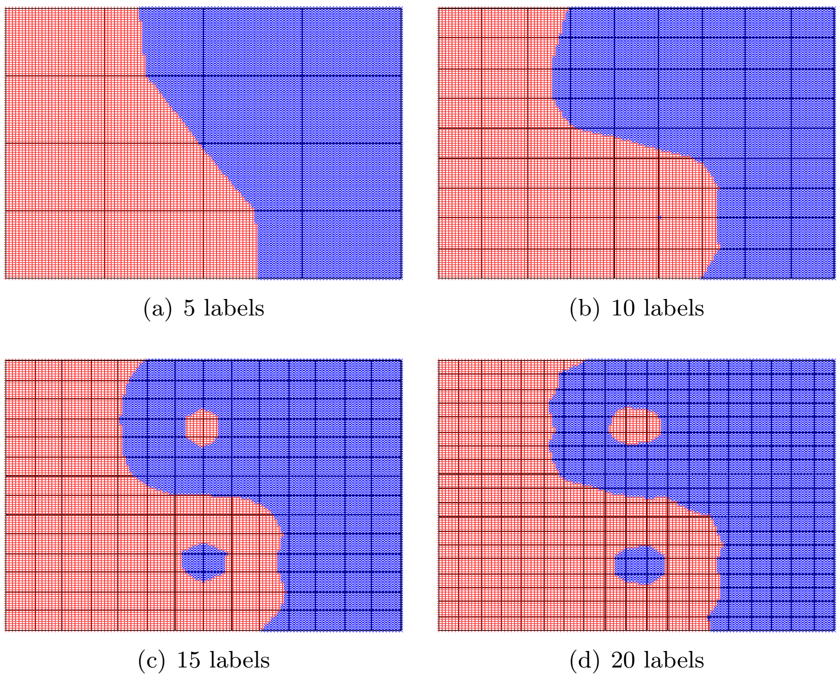
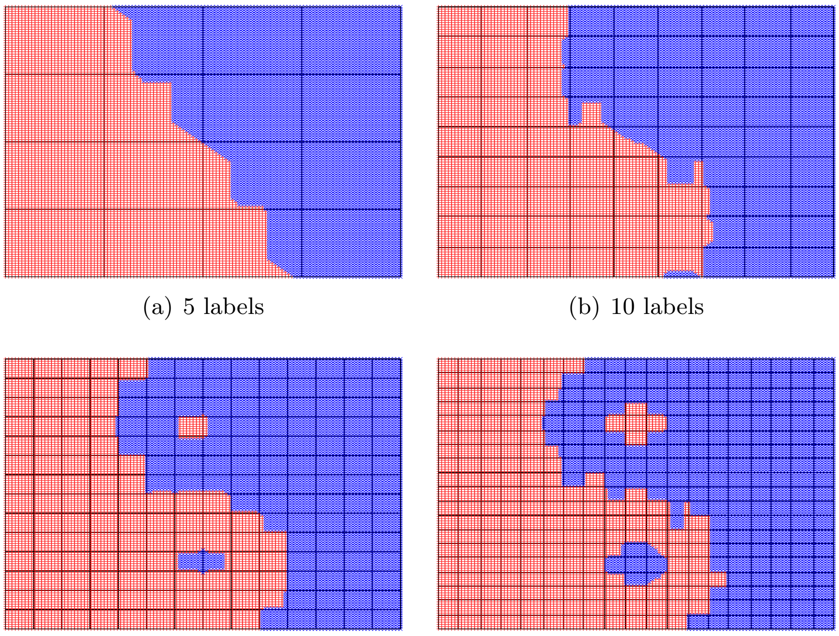

(a) 5 labels (b) 10 labels (c) 15 labels (d) 20 labels

Figure 8.6: Decision boundaries obtained by linguistic Fuzzy-UCS with weighted average inference and 5 (a), 10 (b), 15 (c) and 20 (d) linguistic terms per variable.

5, 10, 15, and 20 linguistic terms per variable; the grid in the plots indicates the partitions in the feature space made by the cross-points of the triangular membership functions associated with the different fuzzy sets. Figure 8.9 shows the decision boundaries obtained by approximate Fuzzy-UCS. Table 8.4 summarizes the training accuracies and population sizes in each case. The results are averages over ten runs with different seeds.

Several observations can be drawn from the evolved decision boundaries. Firstly, the results show the generalization capabilities of all learners. The rules tend to expand as much as possible while they are accurate, covering regions in the feature space where there are no examples. This



(a) 5 labels

(b) 10 labels

(c) 15 labels

(d) 20 labels

Figure 8.7: Decision boundaries obtained by linguistic Fuzzy-UCS with action winner inference and (a) 5, (b) 10, (c) 15, (d) and 20 linguistic terms per variable.

generalization pressure is mostly due to subsumption, which replaces the offspring for most general and accurate rules when possible. Thus, this operator gives highly general and accurate rules more strength.

Interval-based UCS reached the maximum accuracy among all learners. It evolved a population consisting of 1230 rules which accurately defined the decision boundaries (see figure 8.5(b)), with 99.8% training accuracy. The accuracy obtained by linguistic Fuzzy-UCS depended on the number of linguistic terms per variable (see the models built in figures 8.6, 8.7, and 8.8). With 5 linguistic labels per variable, linguistic Fuzzy-UCS could not discover the two inner concepts of the tao problem regardless of the used inference method. The models only defined one linear class boundary that did not fit the curved boundary of the domain accurately. As the number of linguistic terms per variable increased, the boundaries were defined more accurately. With 20 linguistic terms per variable, the three types of inference achieved high training performances.

The models evolved by linguistic Fuzzy-UCS with the three types of inference differed in the shape of the decision boundaries and the rule set size. Weighted average inference defined smooth boundaries which resulted from the vote of several overlapping rules (see figure 8.6). However, it maintained a large number of rules in the final population. Action winner inference created more reduced rule sets, but the boundaries were more abrupt. Note that the decision boundaries followed the partitions produced by the fuzzy membership functions, especially when 15 and 20 linguistic terms were used. This is because only the rules that maximized the product of $\mu_{A^k}(e) \cdot F$ were kept in the final population. Fittest rules inference evolved the most compact rule sets. Furthermore, the boundaries were smoother than the ones obtained with action winner

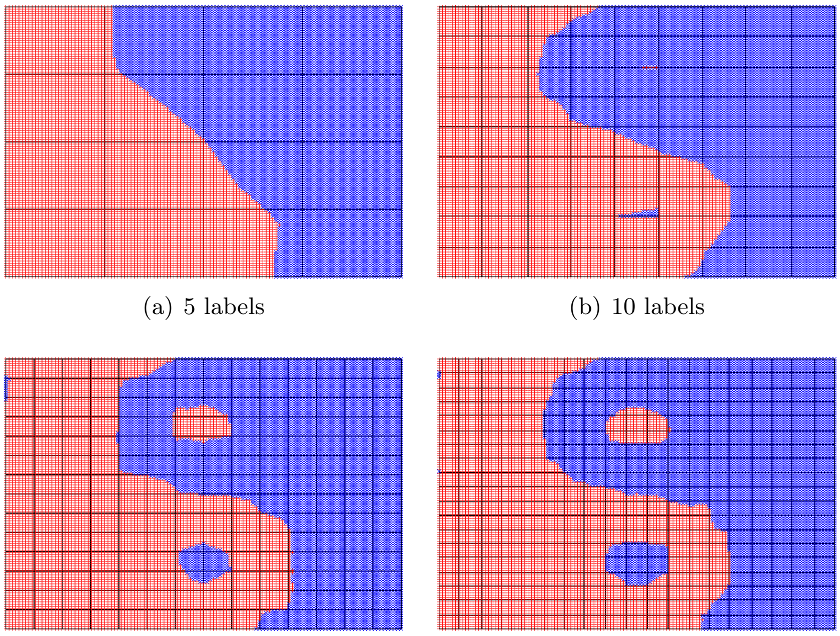

(a) 5 labels (b) 10 labels (c) 15 labels (d) 20 labels

Figure 8.8: Decision boundaries obtained by linguistic Fuzzy-UCS with fittest rules inference and (a) 5, (b) 10, (c) 15, (d) and 20 linguistic terms per variable.

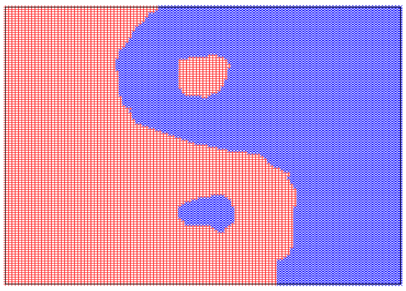


Figure 8.9: Decision boundaries obtained by approximate Fuzzy-UCS.

scheme. This type of inference maintained the most numerous and accurate rules in the final population. As this process could insert overlapping rules into the final population, the weighted average inference was used to infer the class, thus forwarding the interpolative reasoning. For this reason the decision boundaries were not as abrupt as the ones evolved by the action winner inference.

Finally, figure 8.9 shows that approximate Fuzzy-UCS built a model that accurately fitted the training examples. Approximate Fuzzy-UCS used an inference method based on action winner, similar to that used in linguistic Fuzzy-UCS. Regardless of this inference scheme, the decision boundaries were smoother because each variable evolved an independent fuzzy set. However, as a consequence of not sharing a unique semantic, the interpretability of the fuzzy-rules was

Table 8.4: Summary of Fuzzy UCS results with interval-based, approximate and linguistic representation with 5, 10, 15, and 20 linguistic terms per variable in the tao problem. Columns show the training accuracy and the number of rules for action winner and weighted average inference schemes.

	Training acc.			**Num. rules**		
Interval-based UCS	99.80			1230		
App. Fuzzy-UCS	96.94			555		
	wavg	***awin***	***nfit***	***wavg***	***awin***	***nfit***
Lin. Fuzzy-UCS 5L	82.95	83.24	88.31	112	17	15
Lin. Fuzzy-UCS 10L	91.85	91.19	91.85	441	78	30
Lin. Fuzzy-UCS 15L	96.68	94.74	96.68	618	144	52
Lin. Fuzzy-UCS 20L	97.15	95.57	97.15	763	200	65

degraded with respect to the linguistic representation. For example, one of the most numerous rules evolved by approximate Fuzzy-UCS was:

$$\textbf{IF } x_1 \text{ is (-3.3, -1.50, -1.13) } \textbf{ and } x_2 \text{ is (5.50, 6.48, 11.85) } \textbf{THEN } c_1 \textbf{ WITH } w = 0.998, \quad (8.33)$$

where each variable was represented by a triangular-shaped fuzzy set whose vertices could take any possible value in the feature space. Moreover, note that the number of rules evolved by approximate Fuzzy-UCS was larger than those created by any configuration of linguistic Fuzzy-UCS, except for linguistic Fuzzy-UCS with weighted average inference with 15 and 20 linguistic terms per variable. This large number of rules degraded even more the interpretability of the semantic-free approach.

8.5.3 Comparison Between Linguistic and Approximate Representations

This section furthers the study on the three types of inference of linguistic Fuzzy-UCS and compares them to the approximate representation. Specifically, we examine the trade-off between precision and rule set size already pointed out in the previous section for the three types of inference in linguistic Fuzzy-UCS. Besides, we include approximate Fuzzy-UCS in the comparison, which is expected to fit the training data more accurately. Considering the approximate representation, we aim to a) confirm the intuition that the approximate representation permits fitting significantly better the training instances, b) analyze whether this improvement is also present in the prediction of previously unseen instances, and c) evaluate the impact of the approximate representation on the interpretability of the evolved rule set.

Methodology

We selected a collection of 20 real-world data sets whose characteristics are summarized in table 8.1. All the data sets were obtained from the UCI Repository (Asuncion and Newman, 2007), except for *tao*, which was selected from a local repository (Bernadó-Mansilla et al., 2002).

To measure the precision of the method in fitting the training instances, we used the training accuracy rate, i.e., the proportion of correctly classified examples of the training set. The performance of the method was measured by the test accuracy rate, i.e., the proportion of correct predictions on previously unseen instances. To obtain reliable estimates of these metrics, we used a ten-fold cross validation procedure (Dietterich, 1998). We collected the evolved rule set sizes to compare the interpretability of the three configurations of linguistic Fuzzy-UCS. Since the types of rules created by the linguistic representation are different from those of the approximate representation, we qualitatively compared the rule sets built by both approaches.

The results were statistically analyzed following the recommendations pointed out by Demšar (2006). In all the analysis, we used non-parametric statistical tests to compare the results obtained by the different learning algorithms. Parametric tests require that the input data (in our case, the tables of results) satisfy strong conditions, and the tests to check these conditions need large amounts of data (i.e., large number of data sets) to be effective (Sheskin, 2000). For this reason, non-parametric tests are recommended (Demšar, 2006), since they relax the requirements on the input data.

We applied multiple-comparison statistical procedures to test the null hypothesis that all the learning algorithms performed equivalently on average. Specifically, we used the Friedman's test (Friedman, 1937, 1940), a non-parametric equivalent of the repeated-measures ANOVA (Fisher, 1959). If the Friedman's test rejected the null hypothesis, we used the non-parametric Nemenyi test (Nemenyi, 1963) to compare all learners to each other. The Nemenyi test is said to be quite conservative, especially when a large number of learners is compared, so that it might not detect some existent differences between the learners. Therefore, we complemented the statistical analysis by comparing the performance of each pair of learners by means of the non-parametric Wilcoxon signed-ranks test (Wilcoxon, 1945). The approximate p-values resulting from the pairwise analysis, calculated as indicated in (Sheskin, 2000), were provided in the analysis. For further information about the statistic tests, the user is referred to appendix B.

We used the default configuration for Fuzzy-UCS (see section 8.4), since it used equivalent parameter values to those usually set for XCS and UCS. Moreover, we fixed the number of linguistic labels to 5. We did not consider a larger number of linguistic terms since it could hinder the interpretability desired in a linguistic representation. For approximate Fuzzy-UCS, we fixed $r_0 = 1$.

Results

Our first concern was to analyze the precision in fitting the training instances of linguistic Fuzzy-UCS with the three types of inference and approximate Fuzzy-UCS. Thus, we computed the training accuracy obtained with the four approaches, as reported in table 8.5. The two last rows supply the average rank and the position of each algorithm in the ranking. The ranks were calculated as follows. For each data set, we ranked the learning algorithms according to their performance; the learner with highest accuracy held the first position, whilst the learner with the lowest accuracy held the last position of the ranking. If a group of learners had the same performance, we assigned the average rank of the group to each of the learners in the group.

The multiple-comparison test enabled us to reject the null hypothesis that all learners were equally accurate at a significance level of 0.001. Thus, we ran the Nemenyi test at a significance

Table 8.5: Comparison of the training accuracy of linguistic Fuzzy-UCS with weighted average (wavg), action winner (awin), and most numerous and fittest rules inference (nfit), and approximate Fuzzy-UCS on a set of twenty real-world problems.

	Linguistic			**Approximate**
	wavg	**awin**	**nfit**	
ann	99.35	98.34	99.48	98.83
aut	99.30	92.67	98.87	97.82
bal	91.03	90.97	89.97	98.61
bpa	68.57	68.30	69.79	86.29
cmc	67.34	67.77	70.70	65.29
col	93.04	91.76	96.22	98.95
gls	71.04	65.84	71.46	94.46
h-c	89.75	91.11	92.02	98.77
h-s	94.75	92.46	96.70	98.92
irs	95.78	95.59	94.56	97.47
pim	77.05	77.74	79.16	89.91
son	100.00	99.89	99.50	99.91
tao	81.70	83.31	87.42	89.64
thy	89.03	89.92	92.62	95.70
veh	76.77	72.97	77.49	89.52
wbcd	96.38	95.97	96.51	99.69
wdbc	96.34	95.50	96.18	99.55
wne	98.48	97.28	98.12	100.00
wpbc	97.57	94.01	95.39	96.98
zoo	99.71	99.98	99.90	100.00
Rank	*2.70*	*3.45*	*2.40*	*1.45*
Pos	*2*	*4*	*3*	*1*

level of 0.10. Figure 8.10 ranks the four learners and connects those that performed equivalently according to the Nemenyi procedure. The test indicates that approximate Fuzzy-UCS achieved significantly better training performance than all the other algorithms. Moreover, linguistic Fuzzy-UCS with action winner significantly degraded the training performance achieved with Fuzzy-UCS with fittest rules inference. As the Nemenyi test is said to be quite conservative, we also performed pairwise comparisons between the learners by means of the non-parametric Wilcoxon signed-ranks test. Table 8.6 provides the approximate p-values. The $\oplus$ and $\ominus$ symbols indicate that the method in the row significantly improved/degraded the performance obtained with the method in the column. The $+$ and $-$ symbols denote a non-significant improvement/degradation. The pairwise analysis confirmed the conclusions extracted from the Nemenyi test. No other significant differences were found by this statistical test.

As expected, the approximate representation fitted the training examples more accurately since there was no semantic shared among all variables—that is, each variable could define its own fuzzy sets. Next, we analyzed if this improvement was also present in the test performance,

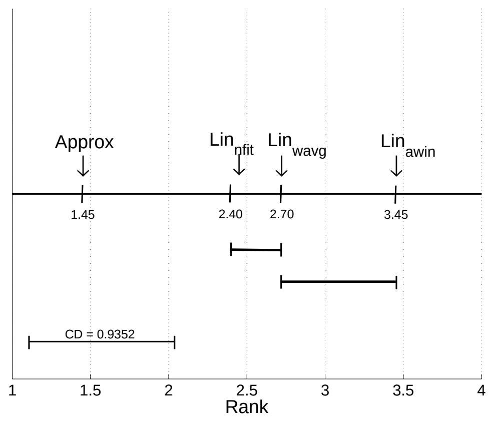


Figure 8.10: Comparison of the training performance of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.10$) are connected.

Table 8.6: Pairwise comparisons of the training accuracy achieved by linguistic Fuzzy-UCS with the three types of inference and approximate Fuzzy-UCS.

	wavg	**awin**	**nfit**	**approx**
wavg		.0793	.0929	.0017
awin	$-$		.0012	.0002
nfit	$+$	$\oplus$		.0017
approx	$\oplus$	$\oplus$	$\oplus$	

which is shown in table 8.7. The multiple-comparison test rejected the hypothesis that all learners performed the same on average at a significance level of 0.001. Figure 8.11 shows the rank of each method and connects the groups of learners that performed equivalently according to the Nemenyi test at a significance level of 0.10. The statistical procedure identified two groups of techniques that performed equivalently. The first group included linguistic Fuzzy-UCS with weighted average inference and approximate Fuzzy-UCS. The second group comprised approximate Fuzzy-UCS and linguistic Fuzzy-UCS with action winner and fittest rules inferences. The same significant differences were found with the pairwise statistical analysis. Table 8.8 supplies the approximate p-values calculated from the Wilcoxon signed-ranks test.

Although the flexibility of the approximate representation allowed Fuzzy-UCS to evolve models that fitted the training examples more accurately, no significant differences were observed in the prediction of previously unseen instances. Moreover, approximate Fuzzy-UCS did not perform as well as linguistic Fuzzy-UCS with weighted average inference, though the difference was not statistically significant. Further analysis pointed out that approximate Fuzzy-UCS was over-fitting the training data in some of the tested domains. To contrast this hypothesis, we monitored the evolution of the training and test performance of the problems in which approx-

Table 8.7: Comparison of the test accuracy of linguistic Fuzzy-UCS with weighted average (wavg), action winner (awin), and most numerous and fittest rules inference (nfit), and approximate Fuzzy-UCS on a set of twenty real-world problems.

	Linguistic			**Approximate**
	wavg	**awin**	**nfit**	
ann	98.85	97.39	98.61	95.44
aut	74.42	67.42	69.32	69.54
bal	88.65	84.40	83.40	82.73
bpa	59.82	59.42	58.93	64.19
cmc	51.72	49.67	49.42	44.79
col	85.01	82.46	78.50	87.96
gls	60.65	57.21	57.43	71.82
h-c	84.39	82.62	82.05	80.16
h-s	81.33	80.78	78.11	78.59
irs	95.67	95.47	93.73	95.80
pim	74.88	74.11	74.32	74.32
son	80.78	73.71	71.66	76.34
tao	81.71	83.02	87.53	89.39
thy	88.18	89.49	91.25	92.28
veh	67.68	65.35	65.34	65.80
wbcd	96.01	95.73	95.29	95.69
wdbc	95.20	94.61	94.51	93.87
wne	94.12	94.86	91.82	95.42
wpbc	76.06	76.05	71.69	59.78
zoo	96.50	94.78	95.90	83.53
Rank	*1.60*	*2.75*	*3.20*	*2.45*
Pos	*1*	*3*	*4*	*2*

imate Fuzzy-UCS degraded the results obtained by linguistic Fuzzy-UCS with any inference type. Figure 8.12 plots the evolution of the training and test performance for the *bal* problem. During the first 5 000 learning iterations, both training and test performance rapidly increased, achieving about 90% and 84% accuracy rates respectively. After that, the training performance continued to increase whilst the test performance slightly decreased. After 100 000 iterations, the training performance reached 98%; nonetheless, the test performance decreased to 82%. Thus, at a certain point in the learning process, the flexibility of the approximate representation led Fuzzy-UCS to over-fit the training instances in order to create more accurate classifiers, which was detrimental to the test performance.

Finally, table 8.9 shows the number of rules evolved in each configuration. Friedman's test rejected the hypothesis that the population sizes were equivalent on average at a significance level of 0.001. The post-hoc Nemenyi test supported the hypothesis that the four learners evolved populations with significantly different sizes. The pairwise comparisons yielded the same conclusions (see in table 8.10 the approximate p-values according to a Wilcoxon signed-

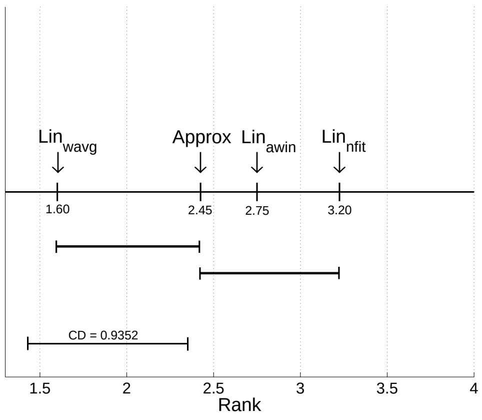


Figure 8.11: Comparison of the test performance of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.10$) are connected.

Table 8.8: Pairwise comparisons of the test accuracy achieved by linguistic Fuzzy-UCS with the three types of inference and approximate Fuzzy-UCS.

	wavg	**awin**	**nfit**	**approx**
wavg		.0032	.0051	.1913
awin	$\ominus$		.2627	.7369
nfit	$\ominus$	$-$		.4781
approx	$-$	$+$	$+$	

ranks test). In fact, a simple quantitative analysis highlighted the differences in the population sizes. Fuzzy-UCS with weighted average inference built populations that consisted of thousands of rules. Consequently, although using a linguistic representation, this large number of rules hampered the interpretability of the rule set. Approximate Fuzzy-UCS resulted in smaller populations; however, these consisted of hundreds of rules. This, together with the loss of interpretability due to the approximate representation, hindered the readability of the rule set. The other two types of inference of Linguistic Fuzzy-UCS, especially the fittest rules inference, resulted in populations with a moderate number of rules. Fuzzy-UCS with fittest rules inference built populations that ranged from tens of to few hundreds of rules.

These results showed the performance-interpretability trade-off in linguistic Fuzzy-UCS already pointed out in the previous section. Weighted average inference significantly outperformed the other two inference schemes since it combined the knowledge of all experienced rules in the final population. As shown in the case study of the previous section, this allowed Fuzzy-UCS to fit complex boundaries even though the fuzzy representation made a discretization of the feature space. Linguistic Fuzzy-UCS could approximate these boundaries by means of evolving a set of partially overlapping fuzzy rules. However, the interpretability of the rule set was degraded by

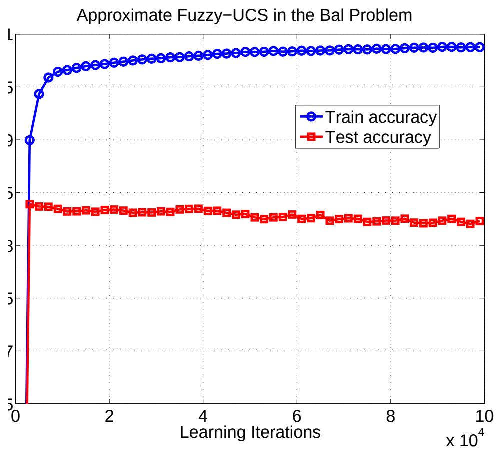


Figure 8.12: Evolution of the training and test accuracies with approximate Fuzzy-UCS on the *bal* problem.

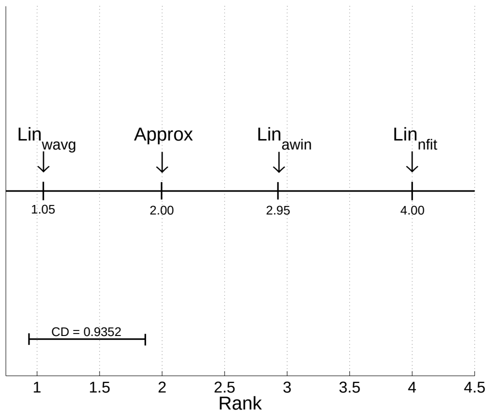


Figure 8.13: Comparison of the number of rules evolved by all learners against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.10$) are connected.

the large number of rules. The other two inference schemes considerably improved the readability, since they produced large reductions of the rule set. Nonetheless, this went against the test performance, which was significantly surpassed by the weighted average inference scheme.

The overall results presented in this section pointed out the viability of the linguistic representation with respect to its approximate counterpart. While approximate Fuzzy-UCS created models that fitted the training data very accurately, there was no statistical evidence of this improvement in the test performance. Furthermore, we also identified that approximate Fuzzy-UCS may over-fit the training instances in complex domains. Thus, the flexibility provided by the approximate representation did not imply an improvement of the test accuracy, although it degraded the readability of the fuzzy-rules. Besides, the rule sets created by approximate Fuzzy-

Table 8.9: Comparison of the population sizes of linguistic Fuzzy-UCS with weighted average (wavg), action winner (awin), and most numerous and fittest rules inference (nfit), and approximate Fuzzy-UCS on a set of twenty real-world problems.

	Linguistic			**Approximate**
	wavg	**awin**	**nfit**	
ann	2769	75	36	409
aut	3872	114	74	158
bal	1212	114	75	441
bpa	1440	73	39	207
cmc	1881	430	271	402
col	4135	154	81	297
gls	2799	62	36	146
h-c	3574	113	46	257
h-s	3415	117	62	231
irs	480	18	7	103
pim	2841	192	62	538
son	3042	178	160	186
tao	111	19	14	464
thy	1283	37	11	134
veh	3732	332	147	532
wbcd	3130	138	28	360
wdbc	5412	276	101	490
wne	3686	95	26	160
wpbc	3772	156	115	175
zoo	773	16	10	55
Rank	*1.05*	*2.95*	*4.00*	*2.00*
Pos	*1*	*3*	*4*	*2*

Table 8.10: Pairwise comparisons of the sizes of the rule sets evolved by linguistic Fuzzy-UCS with the three types of inference and approximate Fuzzy-UCS.

	wavg	**awin**	**nfit**	**approx**
wavg		.0001	.0001	.0001
awin	$\ominus$		.0001	.0001
nfit	$\ominus$	$\ominus$		.0001
approx	$\ominus$	$\oplus$	$\oplus$	

UCS were significantly larger than the ones obtained by linguistic Fuzzy-UCS with action winner and fittest rules inference. For all these reasons, we focus our analysis on linguistic Fuzzy-UCS in the remainder of this chapter, and leave further analysis of approximate Fuzzy-UCS as future research.

8.6 Comparison of Fuzzy-UCS to Several Machine Learning Techniques

So far, we have clearly shown the competitiveness of linguistic Fuzzy-UCS with respect to its approximate counterpart. In this section, we study whether the behavior of linguistic Fuzzy-UCS is comparable to some of the most-used machine learning techniques. For this purpose, we compared Fuzzy-UCS to two sets of learners: fuzzy rule-based learners and "non-fuzzy" (crisp) learners. With the former comparison, we analyzed the behavior of Fuzzy-UCS with respect to other techniques that use the same representation, which may limit the maximum performance that can be achieved in certain domains. With the latter comparison, we study whether, even with the limitations that may impose the fuzzy representation, Fuzzy-UCS is competitive with a large number of the most-representative learners, regardless of the knowledge representation they use. Below, we first present the experimental methodology, and then compare Fuzzy-UCS to the other learners.

8.6.1 Experimental Methodology

The followed methodology is similar to the one presented in the previous section. We selected the same collection of 20 real-world problems, whose characteristics are summarized in table 8.1. The experiments were ran on a ten-fold cross validation, and the test accuracy rate was used to measure the performance of the different learners.

The performance of Fuzzy-UCS was compared with a large variety of learning algorithms, which we organized in two groups. The first group consisted of the following fuzzy rule-based classification systems: Fuzzy GP, Fuzzy GAP, Fuzzy SAP, Fuzzy AdaBoost, Fuzzy LogitBoost, and Fuzzy MaxLogitBoost. Fuzzy GP (Sánchez and Couso, 1998, 2000; Sánchez et al., 2001) is a genetic programming algorithm that builds a fuzzy classifier for each class of the domain by searching for a tree that represents an analytic expression that relates the input and the output variables as accurately as possible. Fuzzy GAP (Sánchez and Couso, 1998, 2000) works similarly to Fuzzy GP, but the optimization system is a hybrid between genetic algorithms and genetic programming. Fuzzy SAP (Sánchez et al., 2001) combines genetic operators with simulated annealing (Korst and Aarts, 1997) to create data models similar to those built by Fuzzy GP and Fuzzy GAP. Fuzzy AdaBoost (del Jesus et al., 2004) is a modification of the boosting algorithm AdaBoost (Freund and Schapire, 1996) to deal with fuzzy rules; Fuzzy AdaBoost generates a compound classifier which decides the output as a linear combination of the outputs of weak classifiers. Fuzzy LogitBoost (Otero and Sánchez, 2006) and Fuzzy MaxLogitBoost (Sánchez and Otero, 2007) are boosting algorithms that iteratively invoke a genetic algorithm to extract simple fuzzy rules that are combined to decide the output of new examples. The basic difference between both algorithms is that Fuzzy MaxLogitBoost may reject a new rule provided by the genetic algorithm if it does not improve the expected global performance. All these methods were run using KEEL (Alcalá-Fdez et al., 2008). We followed the recommended parameter values given in the KEEL platform to configure the methods (Alcalá-Fdez et al., 2008), which also corresponded to the settings used in the bibliography of these methods. We only changed the maximum population size of AdaBoost, LogitBoost, and MaxLogitBoost. We tried population sizes of N={8, 25, 50, 100} for all the data sets, and selected the results of N=50 since they generally allowed us to achieve higher performance ratios than N=8 and N=25, and did not significantly differ from N=100. For all the methods, we used 5 linguistic terms per variable. Fuzzy-UCS was configured as detailed in section 8.5.3.

The second group gathered a large number of learners with different knowledge representations: ZeroR, C4.5, IBk, Naïve Bayes, Part, SMO, GAssist, and UCS. Among them, C4.5, IBk, Naïve Bayes, and SMO have been identified as the top ten data mining algorithms, including supervised and unsupervised learning techniques (Wu et al., 2007). Therefore, the comparison aims at measuring the quality of Fuzzy-UCS with several of the best learners. ZeroR is a simple classifier system that always predicts the majority class in the training data set. We employed this algorithm to provide a baseline result. C4.5 (Quinlan, 1995) is one of the most used decision trees, which derives from ID3 and introduces methods to deal with continuous variables and missing values. IBk (Aha et al., 1991) is a nearest neighbor algorithm; it decides the output of a new example as the most numerous class of the k nearest neighbors. Naïve Bayes (John and Langley, 1995) is a probabilistic classifier that estimates the parameters of a Bayesian model. Part (Frank and Witten, 1998) is a learning architecture that combines the creation of rules from partial decision trees and the separate-and-conquer rule learning technique to create a classifier without using global optimization. SMO (Platt, 1998) is a support vector machine (Vapnik, 1995) that implements the *Sequential Minimization Algorithm*. GAssist (Bacardit, 2004) is a recent Pittsburgh-style LCS. UCS (Bernadó-Mansilla and Garrell, 2003) is a Michigan-style LCS derived from XCS (Wilson, 1995, 1998) and specialized for supervised learning tasks (see chapter 3 for an extensive description of the system). All the methods except for GAssist and UCS were run using Weka (Witten and Frank, 2005). For GAssist, we used the open source code provided in (Bacardit, 2007). For UCS, we used our own code. If not stated differently, all open source methods were configured with the parameters values recommended by default. For UCS we set: $numIter$=100 000, N=6400, acc_0 = 0.99, ν=10, $\{\theta_{GA}, \theta_{del}, \theta_{sub}\}$=50, χ=0.8, μ=0.04, δ=0.1, r_0=0.6. Fuzzy-UCS was configured with standard values as indicated in the previous section.

We applied the following statistical analysis to the results. We used the non-parametric Friedman's test (Friedman, 1937, 1940) to check whether all the learning algorithms performed the same on average. If significant differences were found, two procedures were applied to detect differences among methods. We first aimed at comparing the performance obtained by each of the inference types of Fuzzy-UCS to all other learners (instead of comparing all learners with the others as done in section 8.5). To achieve this, we applied the non-parametric Bonferroni-Dunn (Dunn, 1961) test. Moreover, the analysis is complemented by performing pairwise comparisons among the learners by means of a Wilcoxon signed-ranks test (Wilcoxon, 1945). For further details on the statistical tests, the reader is referred to appendix B.

8.6.2 Comparison to Fuzzy Rule-Based Classification Systems

In the following, we compare the test performance and the interpretability of Fuzzy-UCS with the three types of inference to the aforementioned set of fuzzy rule-based learners.

Comparison of the performance. Table 8.11 details the test accuracies obtained with the fuzzy classifiers Fuzzy AdaBoost, Fuzzy GAP, Fuzzy GP, Fuzzy LogitBoost, Fuzzy MaxLogitBoost, Fuzzy SAP and Fuzzy-UCS with three different types of inference: weighted average (wavg), action winner (awin), and fittest rules (nfit). The average performance of AdaBoost and MaxLogitBoost for the *ann* and *aud* problems is not provided since neither system was able to

Table 8.11: Comparison of the test accuracy of Fuzzy-UCS with weighted average (wavg), action winner (awin), and fittest rules (nfit), to Fuzzy GP, Fuzzy GAP, Fuzzy SAP, Fuzzy AdaBoost, Fuzzy LogitBoost, and Fuzzy MaxLogitBoost.

	GP	GAP	SAP	AdaBoost	LogitBoost	MaxLogitBoost	Fuzzy-UCS		
							wavg	**awin**	**nfit**
ann	77.86	77.20	78.02	-	76.20	-	98.85	97.39	98.61
aut	44.65	45.21	41.00	-	32.63	-	74.42	67.42	69.32
bal	69.73	64.33	65.80	85.54	88.30	75.58	88.65	84.40	83.40
bpa	56.62	57.91	62.30	65.34	64.46	56.53	59.82	59.42	58.93
cmc	47.00	46.57	46.27	49.55	51.10	45.21	51.72	49.67	49.42
col	79.15	73.51	81.89	63.06	63.06	63.06	85.01	82.46	78.50
gls	48.89	47.24	46.42	62.52	68.18	62.18	60.65	57.21	57.43
h-c	73.98	75.09	74.18	60.40	62.09	57.48	84.39	82.62	82.05
h-s	73.70	72.00	72.07	57.56	59.33	57.33	81.33	80.78	78.11
irs	94.47	90.80	91.53	95.47	95.33	92.00	95.67	95.47	93.73
pim	75.32	76.62	77.92	70.69	71.84	72.54	74.88	74.11	74.32
son	64.52	65.99	68.70	46.62	53.38	46.62	80.78	73.71	71.66
tao	80.36	81.75	81.15	91.46	91.73	84.52	81.71	83.02	87.53
thy	86.98	84.94	85.55	97.35	97.08	95.33	88.18	89.49	91.25
veh	46.16	44.59	42.96	30.82	37.25	38.05	67.68	65.35	65.34
wbcd	93.31	92.53	92.72	94.88	94.12	91.83	96.01	95.73	95.29
wdbc	90.93	90.49	91.52	37.26	62.74	37.26	95.20	94.61	94.51
wne	82.91	78.23	79.85	85.59	85.02	77.68	94.12	94.86	91.82
wpbc	74.77	74.47	74.37	23.65	76.35	23.65	76.06	76.05	71.69
zoo	71.18	66.65	66.08	41.89	41.89	41.89	96.50	94.78	95.90
Rank	*5.55*	*6.25*	*5.80*	*5.80*	*4.95*	*7.48*	*2.10*	*3.18*	*3.90*
Pos.	*5*	*8*	*6.5*	*6.5*	*4*	*9*	*1*	*2*	*3*

extract competent fuzzy rules from the two domains, leaving nearly all the feature space uncovered. The authors confirmed that this behavior could be due to the large number of nominal attributes that these two problems have. The last two rows of the table provide the average rank and the absolute position in the ranking of each learner.

The experimental results show that the three configurations of Fuzzy-UCS were the best ranked in the comparison. The next methods in the ranking were the boosting algorithm Fuzzy LogitBoost, the genetic programming-based systems Fuzzy-GP and Fuzzy-SAP, and Fuzzy AdaBoost. Finally, the last methods were Fuzzy GAP, and Fuzzy MaxLogitBoost.

We used the multiple-comparison Friedman's test to analyze whether the differences in the ranking were statistically significant. The statistical test rejected the hypothesis that all the methods performed the same on average at a significance level of 0.001. To evaluate the differences among them, we applied different statistical tests. First, we compared Fuzzy-UCS with each inference type with all the other learners. Figure 8.14 graphically represents the rank of

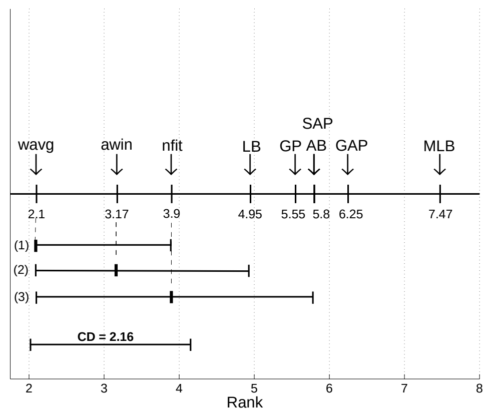


Figure 8.14: Comparisons of one learner against the others with the Bonferroni-Dunn test at a significance level of 0.1. All the learners are compared to three different control groups: (1) Fuzzy-UCS with weighted average inference, (2) Fuzzy-UCS with action winner inference, and (3) Fuzzy-UCS with fittest rules inference. The learners connected are those that perform equivalently to the control learner.

each learner and groups the classifiers that perform equivalently to (1) Fuzzy-UCS with weighted average inference, (2) Fuzzy-UCS with action winner inference, and (3) Fuzzy-UCS with fittest rules inference according to a Bonferroni-Dunn test at a significance level of 0.1. The statistical procedure supported the following hypotheses:

- Using Fuzzy-UCS with weighted average inference as the control learner, the statistical procedure supported the hypothesis that the performance of the control learner was equivalent to the performance of Fuzzy-UCS with the other two inference types. Moreover, Fuzzy-UCS with weighted average outperformed all the other learners.
- Using Fuzzy-UCS with action winner inference as the control learner, the test indicated that this learner performed equivalently to Fuzzy-UCS with the other two types of inference and Fuzzy LogitBoost.
- With respect to Fuzzy-UCS with fittest rules inference, the test did not reject the hypothesis that all the fuzzy learners except for Fuzzy MaxLogitBoost and Fuzzy GAP performed equivalently on average.

As the Bonferroni-Dunn test is said to be quite conservative (Sheskin, 2000), especially when a large number of learners are included in the analysis as in our experimentation, we complemented the statistical study by comparing each pair of learners. Table 8.12 shows the approximate p-values for the pairwise comparison according to a Wilcoxon signed-ranks test. The $\oplus$ and $\ominus$ symbols indicate that the method in the row significantly improves/degrades the performance obtained with the method in the column. Similarly, the $+$ and $-$ symbols denote a non-significant improvement/degradation. The $=$ symbol indicates that each method outperforms and degrades the other in the same number of data sets. Furthermore, figure 8.15

Table 8.12: Pairwise comparison of the test accuracy of fuzzy learners Fuzzy GP, Fuzzy GAP, Fuzzy SAP, Fuzzy AdaBoost (ABoost), Fuzzy LogitBoost (LBoost), Fuzzy MaxLogitBoost (MLBoost), and Fuzzy UCS with weighted average inference (wavg), action winner inference (awin), and fittest rules inference (nfit) by means of a Wilcoxon signed-ranks test.

	GP	**GAP**	**SAP**	**ABoost**	**LBoost**	**MLBoost**	**Fuzzy-UCS**		
							wavg	**awin**	**nfit**
GP		.0366	.2627	.0522	.4115	.0090	.0001	.0001	.0006
GAP	$\ominus$		.2180	.1005	.4781	.0187	.0002	.0002	.0002
SAP	$-$	$+$		.0674	.4330	.0111	.0003	.0005	.0036
ABoost	$-$	$-$	$-$		.0038	.0231	.0045	.0079	.0137
LBoost	$=$	$=$	$=$	$\oplus$		.0003	.0100	.0276	.0438
MLBoost	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$		.0005	.0009	.0007
wavg	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$		.0032	.0051
awin	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\ominus$		.2627
nfit	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\ominus$	$-$	

graphically illustrates the significant differences between methods. That is, each method is depicted in one vertex of the graph, and significant improvements (at α=0.05) of one learner with respect to another are plotted with a directed edge labeled with the corresponding p-value. To facilitate the visualization, Fuzzy AdaBoost and Fuzzy MaxLogitBoost were not included in the graph, since all the other learners significantly improved both methods, except for Fuzzy-GAP, which did not significantly outperform Fuzzy AdaBoost. At a significance level of 0.05, the test indicated that Fuzzy-UCS with weighted average inference significantly outperformed all the other learners, including the two other types of inference of Fuzzy-UCS. Moreover, Fuzzy-UCS with action winner and fittest inference schemes significantly improved the other fuzzy learners, i.e., Fuzzy GP, Fuzzy GAP, Fuzzy SAP, Fuzzy AdaBoost, Fuzzy LogitBoost, and Fuzzy MaxLogitBoost.

Comparison of the interpretability. The study conducted in section 8.5.3 already illustrated the interpretability-performance trade-off among the different inference schemes in Fuzzy-UCS. As shown, the excellent results of Fuzzy-UCS with weighted average with respect to all the other learners were hampered by the large number of fuzzy rules evolved by the method. The other two types of inference appeared as a positive alternative since, although they slightly degraded the accuracy rate with respect to the former approach, they resulted in a moderate number of rules. Aligned with these conclusions, we confirmed the suitability of Fuzzy-UCS by empirically demonstrating that the three schemes of Fuzzy-UCS resulted in significantly more accurate models than those obtained with all the other fuzzy learners. In this section, we further the study and qualitatively analyze if the rule set evolved by these two methods is competitive in terms of readability.

As the type of rules evolved by the systems differ, we qualitatively evaluated the size of the models by extracting some characteristics. Figure 8.16 shows examples of partial models evolved by the fuzzy learners for the *tao* problem. The models built by Fuzzy GP, Fuzzy GAP, and Fuzzy SAP consisted of a rule for each class of the domain. Each rule was directly extracted

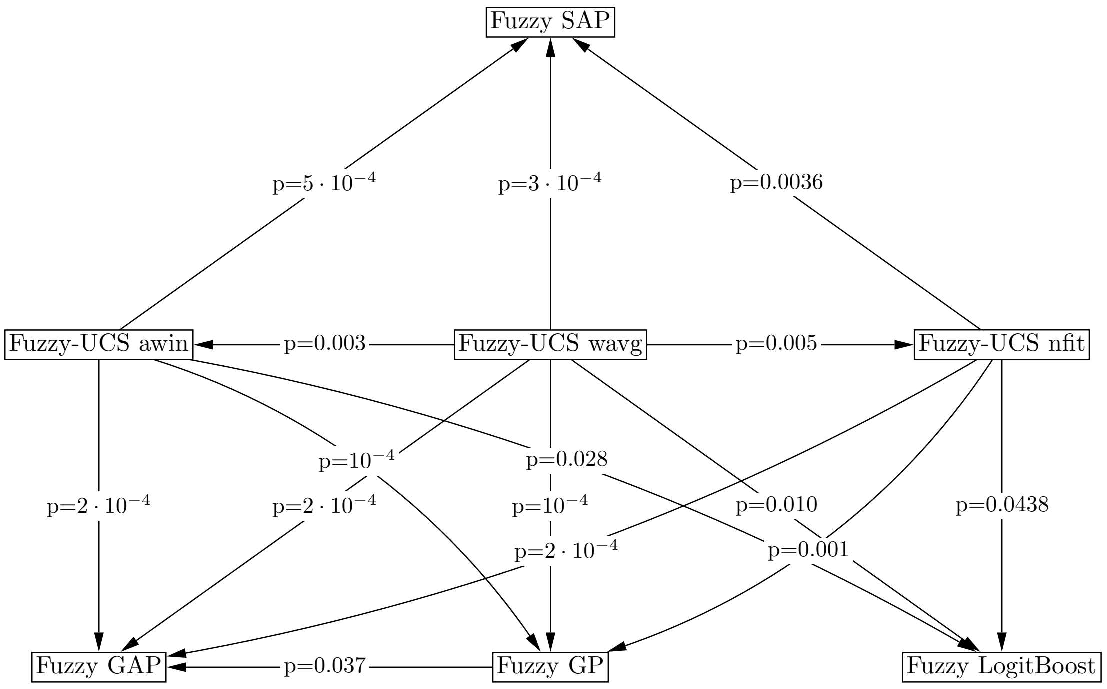


Figure 8.15: Illustration of the significant differences (at $\alpha = 0.05$) of the test accuracy among the fuzzy-methods and Fuzzy-UCS. An edge $L_1 \overset{p_{value}}{\rightarrow} L_2$ indicates that the learner L_1 outperforms the learner L_2 with the corresponding p_{value}. To facilitate the visualization, Fuzzy-AdaBoost and Fuzzy MaxLogitBoost, the two most outperformed algorithms, were not included in the graph.

from an expression codified in a tree. The rules were represented by an arbitrary number of conjunctions (AND) and disjunctions (OR) of conditions over the variables of the domain. One example of these types of rules for a two-dimensional problem is

$$\textbf{IF } (x_1 \textbf{is } \widetilde{A_1^1} \textbf{ AND } x_2 \textbf{ is } \widetilde{A_2^1}) \textbf{ OR } (x_1 \textbf{is } \widetilde{A_1^2} \textbf{ AND } x_2 \textbf{ is } \widetilde{A_2^2}) \textbf{ THEN } \mathrm{c1}, \tag{8.34}$$

where each variable x_i was represented by a linguistic term $\widetilde{A_i} = \{ A_{i1} \vee \ldots \vee A_{in_i} \}$. All variables shared the same semantics which were defined by the combination of triangular-shaped and trapezoidal-shaped fuzzy membership functions (see figure 8.16(a)).

On the other hand, the fuzzy rule-based boosting algorithms created a set of linguistic fuzzy rules that took the following form:

$$\textbf{IF } x_1 \text{ is } \widetilde{A_1} \text{ and } \cdots \text{ and } x_n \text{ is } \widetilde{A_n} \textbf{ THEN } c_1 \textbf{ WITH } w_1^k, \ \cdots, c_m \textbf{ WITH } w_m^k, \tag{8.35}$$

where each variable x_i was represented by a linguistic term $\widetilde{A_i} = \{ A_{i1} \vee \ldots \vee A_{in_i} \}$. All variables shared the same semantics, which was defined by means of triangular-shaped fuzzy membership functions (see figure 8.16(b)). In the consequent part, the rule maintained a weight for each class, which were used for the inference process. Therefore, these individuals rules are less interpretable than the ones of Fuzzy-UCS, which use a single value—the fitness—to infer

```
if y is triangle(-6.0,-3.0,0.0) then red
if ( (x is trapezoid(3.0, 6.0) or x is trapezoid(3.0, 6.0))
     or (x is triangle(0, 3.0, 6.0) or x is trapezoid(3.0, 6.0)) )
   and
       ( (x is triangle ( -3, 0.0, 3.0) or x is trapezoid(3.0, 6.0)
       or . . . )
     ...
then blue
```

(a) GP-based learners

```
if x is L  and y is L   then blue with -5.42 and red with 0.0
if x is M and y is XS then blue with 2.21  and red with 0.0
if x is M and y is XL then blue with -2.25 and red with 0.0
                          ...
```

(b) Boosting learners

```
if x is XL then blue with w=1.00
if x is XS then red  with w=1.00
if x is {XS or S} and y is {XS or S} then red with w=0.87
                          ...
```

(c) Fuzzy-UCS

Figure 8.16: Examples of part of the models evolved by (a) the GP-based methods, i.e., Fuzzy GP, Fuzzy GAP, and Fuzzy SAP; (b) the boosting learners, i.e., Fuzzy AdaBoost, Fuzzy LogitBoost, and Fuzzy MaxLogitBoost; and (c) Fuzzy-UCS for the two-dimensional tao problem. In the fuzzy learners, we used the following five linguistic terms per variable: {XS, S, M, L, XL}. All fuzzy learners use triangular-shaped membership functions. Moreover, GP-based learners also use trapezoid-shaped membership functions.

the class of test instances. The three boosting algorithms supported the absence of a variable by not assigning any linguistic term to the variable. The maximum size of the rule set was a configuration parameter. In our experiments, the maximum population size was set to 50.

To compare these two types of representations to the rule sets evolved by Fuzzy-UCS, we evaluated the size of the models as follows:

- We calculated the size of the models built by Fuzzy GP, Fuzzy GAP, and Fuzzy SAP by counting the number of AND, OR, and IS of the model. This gave us an idea of the average size of the rule. However, note that, due to the flexibility of these types of rules, it was not possible to directly compare them with the rules evolved by the three boosting algorithms and Fuzzy UCS. The rules constructed by Fuzzy GP, Fuzzy GAP, and Fuzzy SAP permit the combination of different logic operators, whose associativity and priority is given by the position of the operators in the tree. An equivalent conjunctive normal form for these rules could be found by applying De Morgan's laws. However, this transformation is not in the scope of this chapter, and so, we only qualitatively evaluated the model sizes.

- The size of the rule sets created by the boosting algorithms and Fuzzy-UCS were computed with the following formula:

$$size = \sum_{i=1}^{N} \frac{1}{\ell} \sum_{j=1}^{\ell} \frac{maxLabels - numLabels(x_i)}{maxLabels - 1}, \tag{8.36}$$

where N is the number of rules in the population, ℓ is the number of variables, and $maxLabels$ is the number of linguistic labels (in our experiments, $maxLabels = 5$). This formula reckons the total number of variables in the model that have, at least, one linguistic term assigned. It also benefits general variables that have more than one linguistic label. To achieve a totally fair comparison, we also referred to the number of rules evolved by Fuzzy-UCS (see table 8.9).

Table 8.13 shows the size of the models created by each fuzzy learner. Table 8.14 illustrates the approximate p-values resulting from the pairwise comparison between the learners according to a Wilcoxon signed-ranks test. For the three methods based on genetic programming, we considered the average number of variables for each rule (i.e., column *is* divided by the number of classes of the problems). The comparison shows that Fuzzy SAP, followed by Fuzzy GP, Fuzzy GAP, and Fuzzy MaxLogitBoost, were the methods that created the smallest models according to a Wilcoxon signed-ranks test at a significance level of 0.05. We have already discussed how the representation of Fuzzy GP, Fuzzy GAP, and Fuzzy SAP was much more flexible and by far less interpretable than the representation of the other learners (see the number of conjunctions and disjunctions with different associativity and priority in the rules). Thus, although the number of attributes per rule was smaller, the interpretability of the model was poor due to the flexibility of the rule (see the partial example provided for the tao problem in figure 8.16(a)). Fuzzy-UCS with weighted average and with action winner inference created the largest and the second largest populations of the comparison respectively. On the other hand, Fuzzy-UCS with fittest rules inference created rule sets that, on average, were not significantly larger than the rule sets built by Fuzzy-GP, Fuzzy AdaBoost, and Fuzzy LogitBoost. Thus, disregarding the three learners based on genetic programming, whose rule sets were poorly readable due to the rule form, only Fuzzy MaxLogitBoost created more reduced populations. However, individual rules of Fuzzy MaxLogitBoost are less interpretable than those of Fuzzy-UCS since they maintain a weight per each class, and all these weights are used in the inference process.

The results provided in this section highlighted the high competitiveness of Fuzzy-UCS in terms of performance and interpretability with respect to other fuzzy learners. In terms of performance, Fuzzy-UCS with any of the three types of inference significantly outperformed all the other fuzzy learners. In terms of interpretability, Fuzzy-UCS with fittest rules inference evolved a number of rules comparable to those evolved by Fuzzy AdaBoost, Fuzzy LogitBoost, Fuzzy GP, Fuzzy GAP, and Fuzzy SAP. In the next section, we broaden the analysis and compare Fuzzy-UCS to a set of general purpose non-fuzzy learners.

8.6.3 Comparison with Non-Fuzzy Learners

Now, we compare Fuzzy-UCS to a set of general-purpose learners that use different knowledge representations: ZeroR, C4.5, IBk, Part, Naïve Bayes, SMO with polynomial kernels of order 3, SMO with Gaussian kernels, GAssist, and UCS. The systems were configured as recommended in the open source implementation, with exception of the following aspects. We ran IBk with

Table 8.13: Size of the models evolved by Fuzzy GP, Fuzzy GAP Fuzzy SAP, Fuzzy AdaBoost (ABoost), Fuzzy LogitBoost (LBoost), Fuzzy MaxLogitBoost (MLBoost), and Fuzzy UCS with weighted average inference (wavg), action winner inference (awin), and fittest rules inference (nfit).

	GP			**GAP**			**SAP**			**ABoost**	**LBoost**	**MLBoost**	**Fuzzy-UCS**		
	and	**or**	**is**	**and**	**or**	**is**	**and**	**or**	**is**				**wavg**	**awin**	**nfit**
ann	30.0	34.4	64.3	27.4	31.4	58.8	5.0	6.8	11.8	-	17.9	-	1038.6	27.2	12.7
aut	27.3	31.8	59.1	30.3	35.5	65.8	5.3	6.9	12.1	-	39.2	-	1555.7	45.1	28.7
bal	27.5	32.8	60.3	21.0	20.2	41.1	4.1	4.5	8.6	19.8	23.8	14.3	578.0	54.3	39.0
bpa	17.6	37.3	54.9	17.9	25.2	43.1	1.8	2.9	4.6	35.3	36.0	13.3	795.5	40.0	19.9
cmc	22.2	25.1	47.2	17.6	18.3	35.9	4.8	3.5	8.3	29.1	27.9	2.0	984.6	223.1	135.8
col	14.6	23.8	38.4	12.2	15.1	27.3	2.1	2.5	4.6	45.0	44.1	0.8	1469.3	50.8	26.0
gls	28.8	32.9	61.7	27.5	29.2	56.7	7.8	7.4	15.2	29.2	31.0	22.8	1293.7	27.8	14.5
h-c	13.6	20.0	33.6	11.4	13.5	24.9	1.7	2.6	4.3	39.6	38.9	1.0	1188.3	34.2	14.2
h-s	16.8	26.1	42.9	8.7	13.2	21.9	2.4	3.4	5.9	40.3	39.2	15.0	1173.0	37.2	18.8
irs	18.7	21.6	40.4	11.5	12.3	23.7	2.5	2.7	5.2	23.4	26.9	4.0	231.2	7.6	2.8
pim	18.3	19.9	38.2	13.7	14.6	28.3	2.0	1.7	3.7	36.9	34.0	13.3	1327.1	86.8	28.0
son	18.2	28.3	46.5	15.1	17.1	32.2	2.0	2.3	4.3	22.3	21.6	0.9	1208.1	70.7	63.4
tao	19.0	20.3	39.2	13.3	19.1	32.4	3.3	3.4	6.7	40.1	43.2	18.0	75.3	12.1	8.6
thy	18.2	20.0	38.1	12.4	13.0	25.4	2.8	2.4	5.1	25.7	29.4	8.8	624.4	16.3	5.0
veh	18.8	21.7	40.4	16.9	18.9	35.8	3.4	4.1	7.4	40.3	37.3	25.6	1641.7	143.8	63.7
wbcd	20.4	40.1	60.5	17.9	20.2	38.1	2.6	3.9	6.5	24.0	27.7	13.1	1033.9	38.9	8.4
wdbc	14.7	15.7	30.4	10.2	12.3	22.5	1.9	2.0	3.9	44.9	43.9	0.9	2108.7	105.4	38.4
wne	15.8	19.5	35.3	14.5	14.9	29.4	2.5	2.7	5.2	30.8	31.2	26.9	1437.7	33.1	9.0
wpbc	24.0	38.1	62.1	11.9	19.2	31.1	2.8	4.6	7.4	44.9	44.0	0.8	1536.6	62.3	45.3
zoo	34.0	34.9	68.9	37.8	38.2	76.0	8.2	9.7	17.9	42.0	36.2	0.9	263.4	5.0	3.3

Table 8.14: Pairwise comparisons of the sizes of the models of Fuzzy GP, Fuzzy GAP, Fuzzy SAP, Fuzzy AdaBoost (ABoost), Fuzzy LogitBoost (LBoost), Fuzzy MaxLogitBoost (MLBoost), and Fuzzy UCS with weighted average inference (wavg), action winner inference (awin), and fittest rules inference (nfit) by means of a Wilcoxon signed-ranks test.

	GP	**GAP**	**SAP**	**ABoost**	**LBoost**	**MLBoost**	**Fuzzy-UCS**		
							wavg	**awin**	**nfit**
GP		.0003	.0001	.0005	.0001	.0137	.0001	.0003	.2179
GAP	⊖		.0001	.0002	.0001	.1671	.0001	.0002	.0228
SAP	⊖	⊖		.0001	.0001	.0276	.0001	.0001	.0001
ABoost	⊕	⊕	⊕		.8666	.0001	.0001	.0793	.1790
LBoost	⊕	⊕	⊕	−		.0001	.0001	.1005	.1084
MLBoost	⊖	−	⊕	⊖	⊖		.0001	.0002	.0105
wavg	⊕	⊕	⊕	⊕	⊕	⊕		.0001	.0001
awin	⊕	⊕	⊕	+	+	⊕	⊖		.0001
nfit	=	⊕	⊕	−	−	⊕	⊖	⊖	

$k = \{1, 3, 5\}$. We ranked the performance obtained by the three configurations, and we only provide the results with the settings that maximized the average rank, that is, $k = 5$ (IB5). The analogous process was carried out for SMO with polynomial kernels. We experimented with polynomial kernels of order $\{1,\ 3,\ 5,\ 10\}$, and supplied the results obtained with polynomial kernels of order 3 since they maximized the average rank. We did not introduce the same system with different configurations in the comparison to avoid biasing the statistical analysis of the results.

Comparison of the performance. Table 8.15 shows the accuracy of the aforementioned learners on the same collection of real-world problems. The two last rows of the table provide the average rank and the position in the ranking of each learner. As proceeds, we discuss several observations that can be drawn from these results.

Firstly, let us highlight the good performance presented by Fuzzy-UCS with weighted average inference. This learner was the third best method in the ranking. Its average rank was really close to UCS, by which Fuzzy-UCS was inspired. Thus, the fuzzy representation did seem not to limit the capabilities of Fuzzy-UCS if all the evolved rules were used to infer the class of new examples. Moreover, the average rank was also close to the best ranked method: SMO with polynomial kernels. The other two inference schemes presented higher average ranks. Fuzzy-UCS with action winner inference and fittest rules inference occupied the 7th and 9th position in the ranking.

We statistically analyzed the results to identify significant differences among the learners. The multiple-comparison Friedman's test rejected the hypothesis that all the learners performed the same on average at a significance level of 0.001. We applied the post-hoc Bonferroni-Dunn test on the results. The test could only reject the hypothesis that the best ranked learners performed equivalently to Fuzzy-UCS with fittest rules inference, SMO with Gaussian kernel, and Zero-R. However, the test has a low discriminatory power for a large number of learners (Demšar, 2006). Thus, we also compared the performance of each pair of learners by means of a Wilcoxon signed-ranks test (see table 8.16). Figure 8.17 uses a graph to illustrate the significant differences between the learners. The test confirmed that Fuzzy-UCS with weighted average inference was one of the best learners in the comparison. It significantly outperformed Naïve Bayes, SMO with Gaussian kernels, ZeroR, and Fuzzy-UCS with the other two types of inference. Moreover, Fuzzy-UCS with weighted average inference did not significantly degrade the results obtained with any other learner. Fuzzy-UCS with action winner inference was only significantly outperformed by SMO with polynomial kernels, and Fuzzy-UCS with weighted average inference. Besides, it significantly improved SMO with Gaussian kernel and ZeroR. Fuzzy-UCS with fittest rules inference presented the poorest results among the three configurations of Fuzzy-UCS. It significantly degraded the results obtained by SMO with polynomial kernels, UCS, IB5, Part, and Fuzzy-UCS with weighted average inference. However, note that it performed equivalently to well-known algorithms such as C4.5, Naïve Bayes, and GAssist.

Comparison of the interpretability. Herein, we qualitatively compare the interpretability of the models created by the different learners. We do not consider IBk, SMO, and Naïve Bayes since their knowledge representation can hardly be compared to the other learners. IBk is a lazy classifier that does not use any knowledge representation; to predict the output of a new input example, IBk searches for the k nearest neighbors and returns the majority class among them. SMO represents the knowledge by $\binom{n_c}{2}$ support vector machines (where n_c is the number

Table 8.15: Comparison of the test accuracy of Fuzzy-UCS with weighted average (wavg), action winner (Awin), and fittest rules inference (nfit) to ZeroR (0R), C4.5, IB5, Part, Naïve Bayes (NB), SMO with polynomial kernels of order 3 (SMO$_{p3}$), SMO with Gaussian kernels (SMO$_{rbf}$), and GAssist.

	0R	**C4.5**	**IB5**	**Part**	**NB**	**SMO$_{p3}$**	**SMO$_{rbf}$**	**GAssist**	**UCS**	**Fuzzy-UCS**		
										wavg	**awin**	**nfit**
ann	76.20	98.90	97.34	98.57	86.33	99.34	91.90	97.88	99.05	98.85	97.39	98.61
aut	32.63	80.94	64.03	74.41	58.79	78.09	45.55	68.63	77.41	74.42	67.42	69.32
bal	45.46	77.42	88.18	82.86	90.57	91.20	88.30	79.57	77.32	88.65	84.40	83.40
bpa	57.99	62.31	58.85	67.56	55.97	59.97	57.99	62.24	67.59	59.82	59.42	58.93
cmc	42.70	52.62	46.51	50.04	50.65	48.75	42.70	53.58	50.27	51.72	49.67	49.42
col	63.06	85.32	81.49	84.51	78.23	75.59	82.41	94.30	96.26	85.01	82.46	78.50
gls	35.65	66.15	64.68	66.62	48.95	66.15	35.65	65.06	70.04	60.65	57.21	57.43
h-c	54.45	78.45	83.16	74.20	82.80	78.59	82.48	80.09	79.72	84.39	82.62	82.05
h-s	55.56	79.26	80.74	80.00	83.33	78.89	82.59	77.67	74.63	81.33	80.78	78.11
irs	33.33	94.00	96.00	94.00	96.00	92.67	93.33	96.20	95.40	95.67	95.47	93.73
pim	65.11	74.23	73.32	74.88	75.80	76.70	65.11	73.76	74.61	74.88	74.11	74.32
son	53.38	71.07	84.05	74.38	69.71	85.52	69.26	75.81	76.49	80.78	73.71	71.66
tao	49.89	95.92	97.14	94.33	80.98	84.22	83.63	91.59	87.00	81.71	83.02	87.53
thy	69.83	94.91	94.85	94.33	97.16	88.91	69.83	92.52	95.13	88.18	89.49	91.25
veh	25.42	71.14	68.91	73.39	46.28	83.30	41.71	67.00	71.40	67.68	65.35	65.34
wbcd	65.52	94.99	97.14	95.71	96.15	96.42	96.13	95.59	96.28	96.01	95.73	95.29
wdbc	63.11	94.40	96.78	94.46	93.13	97.58	92.88	94.24	95.96	95.20	94.61	94.51
wne	39.93	93.89	96.67	93.30	97.19	97.75	39.93	93.19	96.13	94.12	94.86	91.82
wpbc	72.97	71.61	78.85	70.05	69.45	81.25	72.97	72.33	69.40	76.06	76.05	71.69
zoo	41.89	92.81	90.47	93.81	94.47	97.83	76.03	93.97	96.78	96.50	94.78	95.90
Rank	*11.50*	*5.95*	*5.28*	*5.95*	*6.63*	*4.28*	*8.95*	*6.25*	*4.65*	*4.68*	*6.50*	*7.40*
Pos	*12*	*5.5*	*4*	*5.5*	*9*	*1*	*11*	*7*	*2*	*3*	*8*	*10*

Table 8.16: Pairwise comparison of the test accuracy of Fuzzy-UCS with weighted average (wavg), action winner (Awin), and fittest rules inference (nfit) to ZeroR (0R), C4.5, IB5, Part, Naïve Bayes (NB), SMO with polynomial kernels of order 3 (SMO$_{p3}$), SMO with Gaussian kernels (SMO$_{rbf}$), and GAssist by means of a Wilcoxon signed-ranks test.

	0R	**C4.5**	**IB5**	**Part**	**NB**	**SMO$_{p3}$**	**SMO$_{rbf}$**	**GAssist**	**UCS**	**Fuzzy-UCS**		
										wavg	**awin**	**nfit**
0R		.0001	.0001	.0001	.0001	.0001	.0010	.0001	.0001	.0001	.0001	.0001
C4.5	$\oplus$		.7089	.9039	.2627	.4209	.0072	.9405	.2043	.7938	.3905	.0793
IB5	$\oplus$	$=$		.7938	.0534	.4115	.0004	.4553	.8519	.8228	.1084	.0304
Part	$\oplus$	$+$	$-$		.2471	.2959	.0057	.4330	.2180	.4209	.3135	.0251
NB	$\oplus$	$-$	$-$	$-$		.0674	.0333	.1084	.1354	.0333	.1790	.3703
SMO$_{p3}$	$\oplus$	$+$	$+$	$+$	$+$		.0032	.2959	.6542	.1672	.0400	.0366
SMO$_{rbf}$	$\oplus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$	$\ominus$		.0032	.0064	.0004	.0025	.0152
GAssist	$\oplus$	$-$	$-$	$-$	$+$	$-$	$\oplus$		.2180	.5016	.3135	.0859
UCS	$\oplus$	$+$	$=$	$+$	$+$	$-$	$\oplus$	$+$		.4330	.0674	.0366
wavg	$\oplus$	$+$	$=$	$+$	$\oplus$	$-$	$\oplus$	$+$	$-$		.0032	.0051
awin	$\oplus$	$=$	$-$	$-$	$+$	$\ominus$	$\oplus$	$-$	$-$	$\ominus$		.2627
nfit	$\oplus$	$-$	$\ominus$	$\ominus$	$+$	$\ominus$	$\oplus$	$-$	$\ominus$	$\ominus$	$-$	

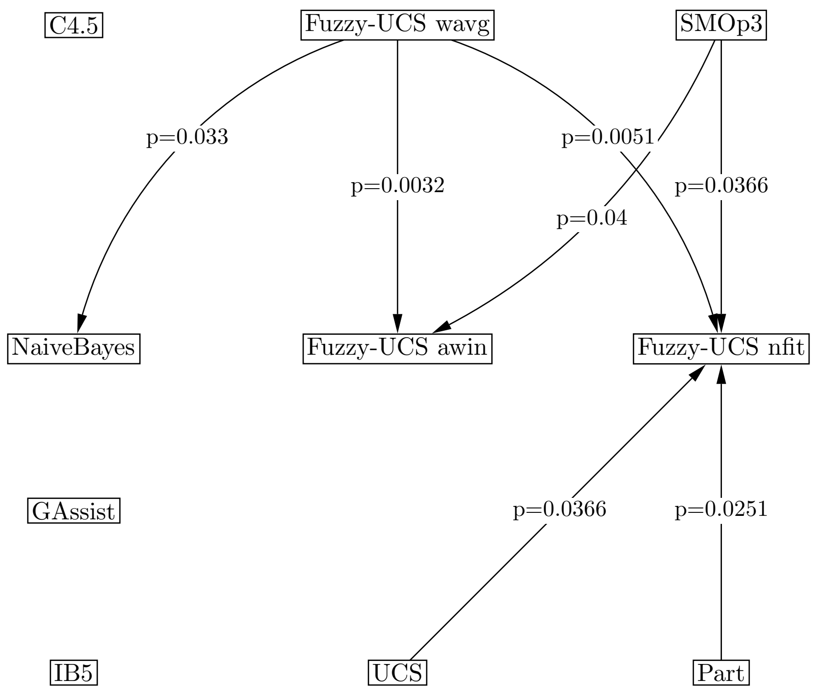


Figure 8.17: Illustration of the significant differences (at $\alpha = 0.05$) of the test accuracy among non-fuzzy methods and Fuzzy-UCS. An edge $L_1 \overset{p_{value}}{\rightarrow} L_2$ indicates that the learner L_1 outperforms the learner L_2 with the corresponding p_{value}. To facilitate the visualization, ZeroR and SMO with Gaussian kernels, the two most outperformed algorithms, were not included in the graph.

of classes), each one consisting of a set of real-valued weights. Therefore, the models created by these two learners are very difficult to interpret. On the other hand, Naïve Bayes builds interpretable models formed by a set of parameters which estimate the independent probability functions and the so-called class-prior of a Bayesian model. Nurnberger et al. (1999) identified a close connection between Naïve Bayes and *neuro-fuzzy classifier systems*, providing a framework that maps a Naïve Bayes classifier into a neuro-fuzzy classifier with the aim of improving its capabilities. The discussion on the difference in the interpretability of Naïve Bayes and their similarity to neuro-fuzzy classifier systems or fuzzy rule-based systems is out of the scope of this chapter. The reader is referred to (Nurnberger et al., 1999) for further details.

Thus, in the remainder of this analysis, we focus on the comparison of the rule-based and tree-based learners, i.e., C4.5, Part, GAssist, UCS, and Fuzzy-UCS. Figure 8.6.3 plots examples of the models evolved by these learners for the two-dimensional tao problem; besides, an example of the weights created by SMO is also depicted. C4.5 evolves trees in which the nodes represent a decision over one variable (see figure 8.18(b)). We evaluated the model size by counting the number of leaves of the tree. Part and GAssist create a set of rules which are defined by conjunction of conditions over their variables, and are interpreted as an ordered activation list (see figures 8.18(c) and 8.18(d)). Moreover, GAssist uses a default rule. UCS evolves a rule set similar to Fuzzy-UCS, but replacing linguistic rules by interval-based rules (see figure 8.18(e)).

```
-   1.000     * <0.229 0.875 > * X]
-   0.298     * <0.708 0.437 > * X]
                   ...
```

(a) SMO

```
x <= -2.75
|  x <= -3.25: red (308.0)
|  x > -3.25
|  |   y <= 1.75: red (55.0)
|  |   y > 1.75
|  |  |  x <= -3: red (11.0/1.0)
|  |  |  x > -3
|  |  |  |    y <= 4.25: blue (6.0)
|  |  |  |    y > 4.25: red (4.0)
                  ...
```

(b) C4.5

```
if x ≤ -3.25 then red (308)
else if x > 2.75 then blue (347/1)
else if y ≤ 0 and x ≥ -1 then red (192/1)
                     ...
```

(c) Part

```
if x > 2.72 and (y is [0.92,4.61] or y > 5.07) then blue
else if ( x is [-0.54, 0.54] or x > 2.72) and y is [-4.28, -2.57] then blue
                                 ...
otherwise red
```

(d) GAssist

```
if x is [-6.00, -0.81] and y is [-6.00, 0.40] then red  with acc= 1.00
if x is [2.84, 6.00]   and y is [-5.26, 4.91] then blue with acc =1.00
if x is [-6.00, -0.87] and y is [-6.00, 0.74] then red  with acc =1.00
                                   ...
```

(e) UCS

```
if x is XL then blue with w=1.00
if x is XS then red  with w=1.00
if x is {XS or S} and y is {XS or S} then red with w=0.87
                               ...
```

(f) Fuzzy-UCS

Figure 8.18: Examples of part of the models evolved by (a) SMO, (b) C4.5, (c) Part, (d) GAssist, (e) UCS, and (f) Fuzzy-UCS for the two-dimensional tao problem.

Each rule can be regarded as an expert classifier in the region of the feature space that it covers. We used the number of rules evolved as the metric of interpretability for Part, GAssist, UCS, and Fuzzy-UCS, although we acknowledge that the measure is not directly comparable as we later discuss. Note that we did not use equation 8.36 to compute the model size because some of the learners are represented by an ordered activation list.

Table 8.17 shows the model sizes of the rule-based and tree-based systems. The number of support vector machines and weights created by SMO and the number of parameters returned by Naïve Bayes are not provided since they can be calculated from the problems characteristics. Qualitatively, it is worth mentioning the following aspects:

- Fuzzy-UCS with weighted average, jointly with UCS, were the two methods in the ranking with higher performance from those that use a rule-based representation. Thus, when performance prevails over interpretability, Fuzzy-UCS is a good approach to face new problems.
- Fuzzy-UCS with weighted average inference, as well as the other two inference schemes, significantly created smaller populations than UCS according to a Wilcoxon signed-ranks

Table 8.17: Average sizes of the models build by C4.5, Part, GAssist, UCS and Fuzzy-UCS with weighted average (wavg), action winner (awin), and fittest rules inference (nfit).

	C4.5	**Part**	**GAssist**	**UCS**	**Fuzzy-UCS**		
					wavg	**awin**	**nfit**
ann	38	15	5	4494	2769	75	36
aut	44	21	7	4565	3872	114	74
bal	45	37	8	2177	1212	114	75
bpa	25	9	6	2961	1440	73	39
cmc	162	168	15	3634	1881	430	271
col	5	9	5	3486	4135	154	81
gls	24	15	5	3359	2799	62	36
h-c	29	21	6	2977	3574	113	46
h-s	17	18	5	3735	3415	117	62
irs	5	4	3	1039	480	18	7
pim	19	7	7	3605	2841	192	62
son	14	8	5	520	3042	178	160
tao	36	17	6	807	111	19	14
thy	8	4	4	1994	1283	37	11
veh	69	32	7	4941	3732	332	147
wbcd	12	10	3	2334	3130	138	28
wdbc	11	7	4	5206	5412	276	101
wne	5	5	3	3685	3686	95	26
wpbc	12	7	4	5299	3772	156	115
zoo	11	8	6	1291	773	16	10

test (at $\alpha = 0.05$). Thus, Fuzzy-UCS achieved one of the main objectives of this work: to create smaller models than those evolved by UCS. Notice that, in addition of evolving smaller rule sets, the individual rules are also more interpretable since the variables are defined by linguistic terms instead of intervals.

- Fuzzy-UCS was the only method in the comparison in which the same semantics (adapted to the universe of discourse of each variable) is shared among all variables, and only 5 linguistic terms were specified. Consequently, Fuzzy-UCS rules were more readable.

The results also indicate that, even Fuzzy-UCS with action winner and fittest rules inferences resulted in moderate-sized populations, the system is still not competitive, in terms of interpretability, with Part, C4.5, and especially with GAssist. However, two important distinctions need to be considered to justify these differences:

- Fuzzy-UCS and, in general, Michigan-style LCSs evolve rules that, by themselves, are experts on the region of the feature space that they cover and collaborate to classify all the input space. Thus, each rule can be regarded as an expert classifier; if the human expert is only interested in a region of the feature space, only the rules involved in this region need to be considered. On the other hand, the rules evolved by Part and GAssist form an ordered activation list. That is, to classify a new example, rules are checked in order and the first rule that matches determines the output. In the case of GAssist, a default rule is used to classify all the examples not matched by any rule in the activation list. This implies that all the previous rules need to be considered to understand why the system is giving this prediction.

- Fuzzy-UCS evolves the rule set incrementally, whilst the other learners go through the data several times to build a model of the data. Incremental learning gives a big advantage to Fuzzy-UCS when learning from large data sets.

The analysis supplied in this section showed that Fuzzy-UCS is highly competitive with respect to a large set of general-purpose machine learning techniques, which include several of the most influential machine learning techniques (Wu et al., 2007). The proposed weighted average version of Fuzzy-UCS was one of the best performers. Thus, a fuzzy rule-based system could achieve accuracy rates as good as—or even better than—other machine learning techniques with knowledge representations that can barely be read by human experts such as support vector machines or instance based algorithms. Moreover, Fuzzy-UCS with the two other inference schemes appeared also to be competitive. Fuzzy-UCS with action winner inference evolved substantially reduced rule sets, although not as much as the ones evolved by GAssist and Part, and it was only statistically surpassed by SMO with polynomial kernels, and our Fuzzy-UCS with weighed average inference. In the next section, we explore the capabilities of the online learning architecture of Fuzzy-UCS to learn from large volumes of data.

8.7 Fuzzy-UCS for Mining Large Data Sets

The two essential differences between Fuzzy-UCS and other rule-based machine learning techniques are that Fuzzy-UCS a) does not perform any form of global optimization, and b) evolves the rule-based knowledge online. Based on a rule set roughly initialized in the first learning iterations by the covering operator, the system is responsible for incrementally evaluating the parameters of the rules and refining the rule-based knowledge by creating more general and more accurate rules. This process provides two main advantages with respect to other learners:

- Fuzzy-UCS learns from a stream of examples. This enables the system to learn from changing environments. This differs from other machine learning methods, such as C4.5, IBk, SMO, and Pittsburgh-style LCSs, which need to process all the training data set in order to produce the final model.

- The learning can be stalled whenever required, and the evolved rule set can be used for predicting the class of new input examples. The more learning iterations the system has performed, the more general and accurate the rules should be. Consequently, the cost of the algorithm increases linearly with the maximum population size N, the number of variables per rule a, and the number of learning iterations n_{learn}

$$Cost_{Fuzzy-UCS} = O\left(a \cdot N \cdot n_{learn}\right), \tag{8.37}$$

but it does not depend directly on the number of examples. In static data sets, it is recommended that n_{learn} be, at least, the number of examples in the training data set.

Table 8.18: Properties of the 1999 KDD Cup intrusion detection data set. The columns describe: the identifier of the data set (Id.), the number of instances (#Inst), the total number of features (#Fea), the number of real features (#Re), the number of nominal features (#No), the number of classes (#Cl), the proportion of instances with missing values (%MisInst), and the dispersion of the data set (Disp) computed as #Fea/#Inst.

Id.	**#Inst**	**#Fea**	**#Re**	**#No**	**#Cl**	**%MisInst**	**%Disp**
kdd'99	494,022	41	35	6	23	0	$8.3 \cdot 10^{-5}$

In this section, we exploit the benefits of online learning in Fuzzy-UCS and apply the system to mine very large data sets. Specifically, we test the performance of Fuzzy-UCS on the 1999 KDD Cup intrusion detection data set (Hettich and Bay, 1999). In the following, we describe the data set and present the experimental results.

8.7.1 Data Set Description

The 1999 KDD Cup intrusion detection data set gathers a large collection of examples of a wide variety of network intrusions simulated in a military environment. We used the subset of 494 022 examples provided in (Hettich and Bay, 1999) that advocate 23 different classes. Each example consists of 41 attributes, which usually characterize network traffic behavior. Table 8.18 summarizes the main traits of the data set.

8.7.2 Results

We ran Fuzzy-UCS on the KDD'99 domain with the default configuration as in the previous section except for $\theta_{GA} = 200$ and $P_{\#} = 0.2$. We increased the period of GA application ($\theta_{GA} = 200$) to permit the classifiers to receive more parameter updates before undergoing a genetic event. We also diminished the probability of generalization in covering ($P_{\#} = 0.2$) since the number of examples per dimension was very high. We ran the experiment for 2 000 000 learning iterations, so that Fuzzy-UCS only received each learning instance an average of 4 times. As in all the experiments performed in this work, to obtain reliable estimates, we employed a 10-fold cross validation methodology (Dietterich, 1998).

Figure 8.19 plots the evolution of the test performance and the population size of Fuzzy-UCS with action winner inference in the first 50 000 learning iterations. That is, we stopped the learning every 2 500 iterations and tested the rule set with the test fold; each dot in the plot corresponds to one of these measurements. Note that the system quickly evolved a highly accurate population. After seeing the first 35 000 examples, that is, a 7% of the whole training data set, the test performance was already about 99%. Increasing the number of learning iterations did not significantly improve the average performance, but it did create more general and equally accurate classifiers. This behavior can be observed in table 8.19, which depicts the test accuracy and the rule set size obtained by Fuzzy-UCS with weighted average inference (1st column), action winner inference (2nd column), and fittest rules inference (3rd column) at different learning iterations. That is, every 500 000 learning iterations, we used the corresponding

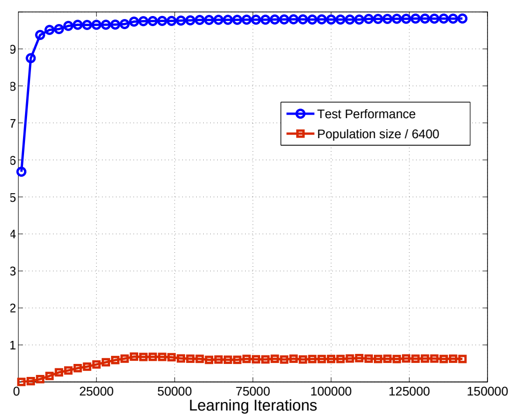


Figure 8.19: Evolution of test accuracies and the population size of Fuzzy-UCS with action winner inference in first 50 000 learning iterations of the 1999 KDD Cup data set.

Table 8.19: Test performance and number of rules evolved by Fuzzy-UCS with weighted average (wavg), action winner (awin), and fittest rules (nfit) in the 1999 Kdd Cup intrusion detection data set at different number of learning iterations.

#Iter	wavg		awin		nfit	
	perf.	**#rules**	**perf.**	**#rules**	**perf.**	**#rules**
500 000	99.32	1944	99.13	541	99.27	417
1 000 000	99.36	2089	99.07	492	99.25	369
1 500 000	99.37	2178	99.02	460	99.24	350
2 000 000	99.36	2257	99.00	428	99.19	323

test set to calculate the accuracy with the three types of inference. While sampling the test examples, all the learning mechanisms of Fuzzy-UCS were disabled, so that the rule set was not modified.

The results show that the number of rules in the final population for the action winner and fittest rules inference decreased as the number of learning iterations increased. Thus, the system was pushing the population toward obtaining maximally general and accurate rules. This behavior was not so clear with the weighted average inference since this inference scheme used all the experienced rules in the final population that have positive fitness, regardless of their generality.

Finally, let us highlight the differences of Fuzzy-UCS with respect to a Pittsburgh-style LCS. In the last section, we configured GAssist with a population of 400 individuals. At the initialization phase, these 400 individuals needed to be evaluated. Thus, the condition of all the rules of each classifier was matched with each of the training instances. This means that, in the population initialization, a Pittsburgh-style LCSs would go through all the data set 400 times, seeing about 180 000 000 instances. The use of windowing mechanisms, as the ones implemented in GAssist, would permit reducing the number of instances that each individual matches to be evaluated by a constant value; nevertheless, note that, in any case, the number of evaluations would increase linearly with the number of input examples. After this, the system would have only the first approximation, and the evolutionary pressures would create new individuals that needed to be evaluated. This makes these types of systems computationally expensive for large data sets. On the other hand, note that Fuzzy-UCS only needed to see 35 000 examples to extract a highly accurate model, and that further iterations were to create a more general rule set. These results emphasize the advantages of online learning, which will be further exploited in future work.

8.8 Summary, Conclusions, and Further Work

The comparative analysis performed throughout this paper has provided many valuable insights on the behavior of Fuzzy-UCS in real-world data sets. As proceeds, we briefly summarize the work. Later, we provide the main conclusions and future work that will be made in light of the promising results supplied in this section. For this purpose, and with the aim of clearly identifying where Fuzzy-UCS is placed in the machine learning community, we perform a SWOT analysis, identifying the strengths, weaknesses, opportunities, and threats of Fuzzy-UCS.

8.8.1 Summary

In this chapter, we proposed a Michigan-style online learning fuzzy-classifier system for supervised learning which iteratively evolves a set of linguistic fuzzy rules which collaborate to cover all the input space. Three schemes of inference and reduction algorithms were designed to infer the output of unknown examples from reduced rule sets. These three mechanisms were designed to offer different levels of rule set reduction and consequently lead to different accuracy rates.

We performed a detailed analysis of the performance and interpretability of the rule sets evolved by Fuzzy-UCS. First, we carefully analyzed the three inference and reduction mechanisms in Fuzzy-UCS with a linguistic representation, and studied if Fuzzy-UCS could generally benefit from the flexibility provided by an approximate representation. This analysis showed that the approximate representation resulted in models that fit the decision boundaries of the problem more accurately. However, it was also detected that this may result in data over-fitting in some real-world domains. In addition, the approximate representation implied a loss of interpretability of the fuzzy rules. All these results supported the use of a linguistic representation.

We also compared Fuzzy-UCS to six fuzzy-rule-based learners and nine general-purpose learners with different types of representations. The analysis showed that Fuzzy-UCS was highly competitive with both groups of learners. In comparison with the fuzzy learners, Fuzzy-UCS with the three types of inference presented the best performances in the study; furthermore, it evolved rule sets with a moderate size. In comparison with the general-purpose machine learning techniques, Fuzzy-UCS with weighted average inference was ranked among the best learners. This showed the suitability of Fuzzy-UCS in spite of the limitations imposed by the discretization of the feature space produced by the linguistic representation. The many benefits of the online fuzzy-rule-based architecture, as well as some drawbacks detected in this study, are detailed in the SWOT analysis of the next section.

Table 8.20: SWOT analysis of Fuzzy-UCS.

Strengths	**Weaknesses**
- It evolves highly accurate models; comparable with the state-of-the-art in classification - It uses a highly legible knowledge representation based on linguistic fuzzy rules - It performs incremental, on-line learning - It is capable of mining large data sets	- It generates moderate or large sized rule sets (depending on the chosen configuration) - Although it can deal with real, integer or categorical features, only application for real and integer data types is recommended
Opportunities	**Threats**
- It may be applied in data streams; further analysis of this problem will be carried out - Because of the use of fuzzy logic, the algorithm could be adapted to deal with vague and uncertain data - The proposed seminar system opens opportunities for further research on fuzzy knowledge representation, the fuzzy inference engine, and evolutionary operators	- A small number of interval-based rules can be interpreted more easily than a large number of linguistic fuzzy rules - Other learning approaches combined with preprocessing can also deal with large data sets

In the final step of the analysis, we exploited the incremental learning architecture of Fuzzy-UCS to extract a model from a large data set: the 1999 KDD Cup intrusion detection data set. It was found that Fuzzy-UCS could quickly evolve a highly accurate model, having seen only the ten percent of the total number of examples in the domain. Incremental learning enabled us to have a rough approximation of the model after a few thousand learning iterations, further refining the rule set as the system received more examples.

8.8.2 SWOT Analysis

All the evidence provided through the experimentations is summarized in the SWOT analysis presented in table 8.20, where *strengths* represent the main advantages of Fuzzy-UCS, *weaknesses* show its drawbacks, *opportunities* outline some suggested further lines of investigation, and *threats* include some optional approaches considered by other methods that could compete with our proposal.

Fuzzy-UCS has four main strengths. First, the system presented high accuracy, which supports the use of Fuzzy-UCS in complex problems. Second, it uses linguistic fuzzy rules, which are much more readable than interval-based rules since all the variables share the same semantics and only a small number of linguistic terms per variable are defined (specifically, in our experiments we only used five linguistic terms per variable). This is really important for domains with high dimensionality where each variable presents different ranges. Third, Fuzzy-UCS is an online process that performs incremental learning, and so, the system neither has knowledge about the data set nor does any kind of global optimization. And fourth, since the run-time complexity of Fuzzy-UCS does not depend on the number of instances in the data set, our system is very useful for mining large amounts of data as we showed with the KDD'99 problem, which consists of about half a million instances, 41 features, and 23 classes.

The main weakness of the system is that, despite the application of reduction schemes, Fuzzy-UCS evolves slightly larger rule sets than those created by other machine learning techniques such as GAssist. Consequently, the number of rules may hinder the interpretability of the evolved knowledge. However, as discussed in previous sections, it is worth highlighting the key differences between the types of rules build by GAssist and Fuzzy-UCS. That is, GAssist does not share any semantics between variables, makes the rules available in an ordered decision list—therefore, each rule depends on the previous rules in the list—, and uses a default rule. Conversely, Fuzzy-UCS evolves independent classifiers that do not depend on the context, and no pressure is applied to evolve default rules. A less important feature of our system is that, although it can work with categorical input variables, fuzzy rules are especially useful for real or integer-valued variables, since in the former case the rule would be equivalent to a classical crisp (or non-fuzzy) one.

We also want to honestly mention some possible threats to Fuzzy-UCS. On the one hand, an expert might find a small number of interval-based rules more legible than many linguistic fuzzy rules (in fact, the degree of interpretability of a system is very difficult to assess when different knowledge representations are compared). On the other hand, there are hybrid learning approaches to deal with problems with large data sets, such as the inclusion of data set reduction techniques, which would allow some of the systems compared in this chapter to address these problems.

Finally, the proposed Fuzzy-UCS algorithm shows some interesting opportunities which will be developed in future work. Firstly, because of its incremental learning capability, the system can be applied to extract information from data streams, which is currently a topic of increasing interest (Arasu et al., 2003; Aggarwal, 2007). Furthermore, the use of fuzzy logics allows the system to be adapted for managing vague and uncertain data, very common in many real-world problems (Sánchez and Couso, 2007; Sánchez et al., 2007). Finally, as future work, we can consider the inclusion of some of the techniques proposed by other systems (such as inference based on an activation list with default rule as in GAssist) and the design of new techniques to achieve greater reductions of the fuzzy rule set without a significant loss of test performance, as well as a more detailed research of other fuzzy knowledge representations.

Chapter 9

Summary, Conclusions, and Further Work

This thesis has investigated *learning classifier systems* as competitive methods for machine learning, addressing two important challenges not only for LCSs, but also for different machine learning techniques: extracting key knowledge from rare classes and building models that are comprehensible to human experts. To address the first challenge, we started a journey departing from a decomposition of the key elements that we would require for any LCS to detect rare classes and finishing with the improvement and application of LCSs to real-world domains with rare classes. To approach the second challenge, we took an inventiveness approach to hybridize the best practices of LCSs, GAs, and fuzzy systems; this resulted in the first Michigan-style learning fuzzy-classifier system for classification tasks.

In this chapter, we first summarize the results and critical observations provided along the consecution of each objective. Thereafter, in addition to the particular conclusions derived and contributions provided under each case, we abstract the work and make an effort to highlight the lessons learned on the way. Finally, we briefly discuss future work that will be made in light of the insights and results provided by this thesis.

9.1 Summary and Conclusions

The present work started with the identification of Michigan-style LCSs as mature machine learning techniques for which we had (1) competent implementations, (2) theory for design, and (3) applications in important domains. In addition, we discussed three main characteristics that make Michigan-style LCS an appealing alternative for machine learning. That is, Michigan-style LCSs (1) evolve a distributed solution in parallel, searching at different pace in different regions of the search space, (2) create a set of independent classifiers, and (3) learn the model online from a stream of examples. Among them, we highlighted the added value provided by their online learning architecture, which makes them suitable to deal with current scientific and industrial applications in which data are usually generated online and models need to be learned on the fly. With these potential advantages of Michigan-style LCSs in mind, we proposed to address the following two critical challenges in machine learning with LCSs:

1. Learning from domains that contain rare classes.
2. Building more understandable models and bringing reasoning mechanisms closer to human ones.

These challenges were studied in the context of two particular LCSs: XCS and UCS. We selected XCS since it is, by far, the most influential Michigan-style LCS. We also included UCS as this thesis was especially concerned with classification problems, and UCS is a specialization of XCS for supervised learning. The inclusion of the two LCSs set the first objective of this work. That is, UCS was identified as a young system that had received no research since its initial design in 2003, and some open issues such as the suitability of the fitness computation scheme, which did not share resources, remained unaddressed. In addition, the advantages of the new architecture of UCS with respect to XCS in classification problems needed further investigation. For this reason, we first proposed to update UCS and compare it with XCS on a set of boundedly difficult classification problems. Then, taking XCS and the new version of UCS as starting point, we focused on the challenges of modeling rare classes and building more comprehensible classification models. More specifically, we articulated the following four objectives:

1. Revise and update UCS and compare it with XCS.
2. Analyze and improve LCSs for mining rarities.
3. Apply LCSs for extracting models from real-world classification problems with rarities.
4. Design and implement an LCS with fuzzy logic reasoning for supervised learning.

Below, we summarize the work done under each objective and provide the key conclusions extracted from each point.

Revise and update UCS and compare it with XCS. The first objective of this thesis proposed the design of a fitness-sharing scheme for UCS, since resource sharing schemes have been shown to provide benefits to both GAs and LCSs. Therefore, we followed a creative analysis to design a fitness-sharing scheme for UCS based on those of XCS and other Michigan-style LCSs. With the introduction of the fitness-sharing scheme, the parameter update procedure of UCS was slightly modified. To evaluate whether the new scheme was beneficial to UCS, we compared the original UCS with UCS with fitness sharing on a collection of four boundedly difficult problems. Later, XCS was included in the comparison with the aim of highlighting the differences between both LCSs in classification tasks.

The empirical analysis provided several insights into the advantages of fitness sharing and into the differences between XCS and UCS. On the one hand, fitness sharing, while not being prejudicial in any problem, appeared to be crucial in domains where there was a tendency to create over-general classifiers. In these problems, the new fitness-sharing scheme enabled UCS to make a stronger pressure toward the removal of over-general classifiers when the first accurate classifiers were discovered. On the other hand, the analysis also made evident that the specialized architecture of UCS enabled the system to solve the tested problems with less computational resources than those required by XCS. And finally, as UCS computes classifiers' accuracy as the true proportion of correct predictions of the classifier instead of as the accuracy of the payoff prediction, the system produced a more reliable fitness pressure toward the optimal population than XCS. These results were expected as UCS is specialized for supervised learning tasks, whereas XCS is a more general architecture that can be applied to reinforcement learning. Thence, the experimental results confirmed that the architecture of UCS is better suited than the XCS's one for classification problems.

Analyze and improve LCSs for mining rarities. With XCS and the updated version of UCS, we faced the problem of mining imbalanced domains with LCSs. Instead of developing new approaches that may improve the system performance in particular domains—but provide little understanding of the real problems that may affect LCSs—we took a more general approach. We abstracted the problem and considered any Michigan-style LCS as a system that evolves a distributed collection of sub-solutions—or niches, which, in turn, contain classifiers—that are evaluated and created online. Thence, we considered the problem of having different distributed sub-solutions—some of which are activated with a lower frequency—that compete for the environmental resources. Then, we followed a design decomposition approach and defined five elements, concerning the creation and evaluation procedures, that needed to be guaranteed to efficiently and scalably solve class-imbalanced problems.

The elements of the design decomposition were analyzed for the particular cases of XCS and UCS. Facetwise models were derived to explain the different elements, and critical conditions for convergence were detected for both systems. More specifically, the study included the following five points:

- The parameter update procedures were examined, providing key recommendations on how they should be configured to ensure an accurate estimation of the parameters of over-general classifiers.
- The capabilities of the covering operator to generate classifiers representing rare classes were analyzed.
- Population size bounds were derived to guarantee that both XCS and UCS could sustain niches that are infrequently activated.
- The effect of the period of activation of the GA on the preservation of infrequent solutions was studied.
- Takeover time expressions for proportionate and tournament selection were deduced, and critical conditions of niche extinction were developed in the same analysis.

In summary, for XCS, we theoretically demonstrated and empirically validated (1) that the parameter update procedure needs to be set inversely proportional to the imbalance ratio to ensure accurate estimates of classifier parameters[1] and (2) that either the population size or the period of application of the GA needs to increase linearly with the imbalance ratio to warrant that an accurate model can be extracted from the under-sampled class. Also, conditions where no convergence can be guaranteed were predicted by the niche extinction models. For UCS, the same conclusions could be extracted, with two key differences. First, the parameter update procedure of UCS was more robust than that of XCS and did not need any especial configuration to deal with domains with large imbalance ratios. Second, the theory developed

[1]In particular, we showed that, with the Widrow-Hoff rule, β needs to decrease linearly with the imbalance ratio to ensure reliable estimates of over-general classifier parameters

for UCS predicted an upper bound on the system behavior. All these analyses resulted in key insights, which were articulated as configuration recommendations, enabling both XCS and UCS to solve highly imbalanced problems that previously eluded solution.

Overall, the main conclusion derived from this objective is that LCSs are competitive machine learning techniques that can effectively solve problems with rare classes, scaling linearly with the imbalance ratio. Therefore, this supports that LCSs are ready to tackle real-world problems with rarities.

Apply LCSs for extracting models from real-world classification problems with rarities. Facetwise analysis provided key insights about LCSs behavior on domains with class imbalances, pointing toward several recommendations that need to be followed to learn accurate models from rare classes efficiently. These models proposed to configure both XCS and UCS according to the imbalance ratio of the training data set, which was assumed to map the imbalance ratio among starved and nourished niches. Nonetheless, this information is not available in real-world domains with unknown characteristics, which opens a gap between the recommendations provided by the theory and its application.

Therefore, under the current objective, we first started to bridge the gap between theory and application in real-world domains. For this purpose, we redefined the concepts of niche, representative classifier, and over-general classifier for domains with continuous attributes. This also led to the redefinition of the problem of mining rare classes. In the interval-based representation, we detected that the niche formation depended on the combination of the expressiveness of the knowledge representation and distribution of examples in the feature space. In brief, we showed that starved niches—or small disjuncts, as termed in the machine learning community—could appear in completely balanced data sets if the form defined by the conditions of the classifiers—in our case, hyper rectangles—did not match the shape of the class boundary. For example, the hyper rectangular representation required the creation of starved niches to accurately approximate oblique class boundaries, regardless of the imbalance ratio of the training data set. Thence, this observation evidenced the relevance and broadness of the problem with starved niches, which can appear in any real-world problem.

Therefore, we identified the new problem of estimating the ratio of the frequency of activation of nourished niches to the frequency of activation of starved niches to enable the application of the recommendations extracted from the theoretical analysis. For this purpose, we designed a heuristic procedure that automatically computes the niche imbalance ratio and self-adapts both XCS and UCS. Empirical results showed that the self-adaptation mechanisms enabled XCS and UCS to solve problems with rare classes without being informed of the actual imbalance ratio.

With the new heuristic procedure, XCS and UCS were ready to face real-world problems, self-adapting themselves according to the information gathered during the evolution. Then, the objective was to analyze whether the two LCSs were competitive with respect to other machine learning techniques in extracting classification models from domains with rare classes. For this purpose, we compared the accuracy of XCS's and UCS's models with those created by three of the most influential machine learning techniques: C4.5, SMO, and IBk. The empirical results showed that both LCSs provided the most accurate models on average. In addition, we extended the comparison by considering four of the most-competent re-sampling techniques: random over-sampling, under-sampling based on Tomek links, SMOTE, and cSMOTE. In general, random over-sampling and SMOTE enabled the learners to create more accurate models of the minority class.

In addition to the observations pointed out in the experimental analysis, the whole comparison came with a crucial conclusion: the performance of each learning system—and each re-sampling technique if used—depended on each particular problem. That is, although general conclusions could be extracted, such as that XCS was the classifier that obtained the most accurate models in the majority of the problems, this does not guarantee that the best method on average will be the best performer for a new real-world, unknown problem. Actually, along the analysis, in several particular cases, the worst method of the comparison provided the most accurate models or vice versa, or re-sampling techniques that resulted in poorer results than those obtained with the original data sets were identified. Therefore, this conclusion demands further studying the characteristics that make real-world problems difficult to learn for each specific learner. This future line work is discussed in more detail in the following sections.

Design and implement an LCS with fuzzy logic reasoning for supervised learning. After analyzing and improving LCSs with the aim of evolving more accurate models of rare classes, the last objective of this work pointed toward increasing the interpretability of the models evolved by LCSs and using reasoning mechanisms that are similar to the human ones. Providing understandable models is a key aspect in supervised learning, especially in critical domains where human experts may require an explanation of the prediction to contrast it with their knowledge, thoughts, or beliefs. For example, in medical domains, human experts may require an explanation of a given diagnosis to see if it confirms or refutes their initial diagnosis.

While in the previous objectives we took an analytic approach to studying existing systems, we followed a creative analysis to achieve the last objective of the thesis. That is, we mixed the ideas that come from LCSs—which provide an accurate online evaluation system—, GAs—which represent a robust discovery component—and fuzzy logic—which supplies human-like representations and reasoning mechanisms. The combination of the three ideas was not novel itself, but the way in which the elements were combined was. The result was the first Michigan-style fuzzy-classifier system that evolves a linguistic fuzzy rule set online for classification tasks. It is worth noting that the system was not a result of a trial/error process, but was derived from a careful analysis of the different ways in which fuzzy logic could be introduced in both the representation and the reasoning mechanisms of UCS. The presented learning method resulted from few iterations on the initial design.

The system was provided with two important characteristics that came from the crossbreeding between LCSs and fuzzy logics, which prepare the system to deal with new challenging problems. Inherited from Michigan-style LCSs, Fuzzy-UCS consists of an online learning architecture; therefore, it can learn from data streams (Aggarwal, 2007; Gama and Gaber, 2007). Thence, the system can deal with applications in which a continuous flow of labeled data is generated, which is usual in many current industrial applications. On the other hand, due to the integration of fuzzy logic into the system, the evolved linguistic fuzzy rules are more legible, and the reasoning mechanisms are closer to human reasoning. Moreover, rule reduction mechanisms were designed for each fuzzy inference. Therefore, the system is prepared to deal, effectively and scalably, with problems with vague and uncertain data, which is very common in many real-world problems (Sánchez and Couso, 2007; Sánchez et al., 2007).

Fuzzy-UCS was extensively analyzed. As Fuzzy-UCS was applied to real-world problems with unknown characteristics, we evaluated the quality of the evolved models by comparing them with the models created by several of the most influential machine learning techniques for pattern recognition. The experimental results showed that Fuzzy-UCS statistically outperformed all the fuzzy learners included in the comparison; also, Fuzzy-UCS was one of the best methods when compared to non-fuzzy learners. Besides, the models evolved by Fuzzy-UCS were, by far, more interpretable than the correspondent models in UCS. Finally, the advantages of the hybridization of LCSs and fuzzy logics was shown by using Fuzzy-UCS to solve a very large problem, the 1999 KDD Cup intrusion detection data set.

In general, the contributions of this work emphasize that Michigan-style LCSs represent a general-purpose, highly-competitive framework for the evolution and the discovery of independent classifiers online. The results provided along this work showed that this framework resulted in competent implementations, that is XCS and UCS, that can capture and accurately model rare classes on the fly. Furthermore, we also illustrated that LCSs easily permit the crossbreeding of ideas, enabling the integration of the best practices that come from several fields. All this makes LCSs one of the most appealing alternatives for facing new challenging applications, which usually require the discovery of knowledge from rarities, the integration of complex representations, and the combination of new techniques. For this reason, LCSs probably have much to say in the future of machine learning.

In addition to the conclusions extracted from the contributions of this work, the development of the present thesis also resulted in valuable lessons for facing new engineering and machine learning problems. The next section discusses these lessons in more detail.

9.2 Lessons from LCSs Design and Application

On our way toward the thesis objectives, two notably different strategies have been followed. While we employed an analytical approach to study LCSs in domains with rare classes, we took an inventiveness methodology and creative analysis to build a machine that mixed the ideas coming from the LCSs, the GAs, and the fuzzy logic fields. Here, we raise the analysis one notch and emphasize the general lessons learned from the application of the two approaches to solve the problems that were defined in the beginning of this document. More specifically, we want to emphasize the following two key lessons:

1. The importance of design decomposition.
2. The relevance of ideas crossbreeding for successfully dealing with complex applications.

These two lessons are further discussed in what follows.

The importance of design decomposition. When we started to approach the challenge of learning from imbalance domains with LCSs, we faced the problem of improving a complex existing learning architecture in a complicated problem. At that point of the thesis, we initially thought about the following three main methodologies to address the problem:

1. Follow intuition to design new modifications that may help discover rare classes.

2. Develop models of the whole system.
3. Decompose the problem in critical elements and analyze them separately.

The first two approaches provide two completely opposite ways of focusing the problem. Using intuition to develop new approaches may lead to some modifications of a particular system that may help get more accurate models of rare classes. Actually, this approach was initially followed in (Orriols-Puig and Bernadó-Mansilla, 2005a,b, 2007). Although this resulted in some improvements on LCSs behavior, these works could not explain the real problems that LCSs may need to face when learning from domains with rare classes. On the other end of the spectrum, we have the option of building global models of the system and then using these models to study the impact of rare classes. The main difficulty of this approach is that the models have to consider all the interactions among the system components. Thence, this may require a high cost to derive complex mathematical equations, especially in LCSs, where several components interact during learning. Moreover, the complexity of the equations themselves may hamper some key insights.

In this thesis, we took the third approach and followed a design decomposition methodology. That is, we sought an effective design decomposition for the problem of creating accurate models of rare classes with LCSs. This lead us to a decomposition in five elements that focused on the problems that LCSs may have in discovering starved niches, and the study of each element permitted us to derive simple models that provided key insights into the system behavior and served as a design tool to improve LCSs' ability to solve highly imbalanced domains. The benefits of this approach have been soundly evidenced along the present work.

Therefore, the lesson extracted here is that design decomposition is a powerful tool not only for analyzing GAs and LCSs, but also for studying any complex system for which models of the whole system are too complex or costly. That is, design decomposition proposes a novel engineering methodology in which complex problems are analyzed by decomposing them into simpler subproblems and, with little algebraic effort, facetwise or "little" models are derived for each subproblem assuming that all the other facets behave in an ideal manner. The models of different facets provide key insights on the system behavior and can be used as a tool for designing new competent systems that satisfy the requirements identified by the models. Furthermore, this approach enables us to save efforts in computing more complex models that, sometimes, may hide some important insights in their complexity. This thesis has provided a good example of how "little" models can help us increase our understanding of complex systems, solving new challenging problems that previously eluded solution. Therefore, the results of this work provide another example of the power of design decomposition, which we hope help encourage other researchers to follow this approach—or even utilize some of the derived models—to analyze complex systems and apply LCSs to new challenging problems.

The relevance of ideas crossbreeding for successfully dealing with complex applications. The second important conclusion of the present work is the need of mixing ideas that come from different learning paradigms to tackle, scalably and efficiently, increasingly complex real-world problems and to build more robust, intelligent, and practical machine learning techniques. That is, current real-world applications usually consist of a large number of examples with complex structures and, sometimes, with vagueness and uncertainty. Most of the typical learning methods are not ready to deal with these types of data or, simply, are not scalable and so not applicable to large data sets. Therefore, this situation demands gathering the best practices of different machine learning fields and building scalable learners that take the best characteristics of each area. Actually, fields such as *soft computing* were born with the aim of crossbreeding the ideas that come from different machine learning areas.

Fuzzy-UCS is an example of fruitful crossbreeding of LCSs, GAs, and fuzzy logics which resulted in a system that can evolve fuzzy rule sets online from a stream of data that may contain vagueness or uncertainty. In addition to the benefits shown in the application of Fuzzy-UCS across several problems, its design comes with the second key lesson of this thesis. That is, the design of Fuzzy-UCS makes evident that LCSs are a "*friendly*" framework for ideas crossbreeding, probably because they were initially designed as a general framework to create artificial intelligence itself. If we regard the general model proposed by the several Michigan-style LCSs studied in the present work, we can realize that LCSs propose a largely general learning model that permits easily incorporating new knowledge representations, apportionment of credit algorithms, or even new search procedures. Therefore, in the current machine learning era, where the field is constantly faced with increasingly challenging problems and ideas crossbreeding appear as the best way to tackle these problems, LCSs turn to be one of the most appealing methods to build next generation machine learning techniques.

All the results, the conclusions, and the lessons learned in this work have also served to fix new objectives that will be approached in further work. In the next section, we take a glance ahead and define two major future research lines that derive from this thesis.

9.3 Further Work

Along the consecution of the objectives of this thesis, several open issues that demand further research have been identified in the end of each chapter. Here, we discuss two important future work lines that come directly or indirectly from the work of this thesis. These two new goals that will guide our immediate future work are:

1. Design measures to characterize real-world classification problems and relate this characterization to the theory.
2. Adapt LCSs to extract association rules online.

The former, the need for a better characterization of real-world classification problems, was already emphasized in the comparison of several machine learning techniques performed in chapter 7. The later, the adaptation of LCSs to extract association rules online, comes motivated by the types of problems in industry that currently face machine learning techniques. The motivation for each of these two lines is elaborated in more detail in what follows.

Design measures to characterize real-world classification problems and relate this characterization to the theory. The study of the class-imbalance problem in LCSs indirectly evidenced the increased difficulty of dealing with real-world problems. That is, in the first stage of the study, we took an analytical approach, derived facetwise models, and validated these models with artificial problems in which we could control the different dimensions of problem difficulty. The experiments emphasized that, given the particular complexities of the artificial problems, the theory could explain the behavior of LCSs on these problems. Nonetheless, the same analysis could not be directly applied to real-world problems. That is, the fact that the characteristics of real-world problems were unknown opened a gap between the theory and its application.

Therefore, to evaluate LCSs performance, we compared them with a collection of top-notch machine learning techniques and analyzed which learners resulted in the most accurate results on average. This type of comparison permitted us to extract conclusions on the average performance of the learners; however, it provided poor information about whether a given learner would be the best performer in a new real-world problem. This was an inevitable effect of having no information about the intrinsic complexities of the new problem. Therefore, in general, we could barely predict whether a given learner can efficiently solve a new problem. To do this, we need to (1) identify the sources of complexity that affect each particular learner and (2) create some methodology to characterize real-world problems. While the first types of analyses are common in machine learning (the facetwise analysis provided in this thesis is a clear example), the characterization of real-world problems is still a young field.

Consequently, the first future work line aims at advancing in the characterization of real-world problems. For supervised learning, Ho and Basu (2002) designed a collection of metrics that provide some indicators about the geometrical characteristics of the class distributions in the training data set. Some analyses that relate these metrics with the error of XCS have been already conducted (Bernadó-Mansilla and Ho, 2005; Bernadó-Mansilla et al., 2006). In spite of these first successful analyses, these complexity metrics were not enough to fully capture all the sources of difficulty of classification problems. Thence, as further work, we will follow up these works, taking the defined metrics as starting point, with the aim of defining more metrics and using them to characterize real-world problems. This characterization would permit us to approach the theory to the actual difficulties of real-world problems.

Moreover, it appears appealing to use these metrics to enrich the study of different learning systems by comparing them on collections of data sets with a certain given complexity. That is, the characterization of learning domains would permit identifying the problem characteristics for which each learner is better suited than the others, thence, identifying the sweet spot where each learning algorithm is the best in the comparison. Furthermore, the definition of complexity metrics would enable us not only to assess the difficulty of existing data sets, but also to create new data sets with certain complexities to evaluate learning methods. The first steps toward this promising research area have been taken in (Macià et al., 2008a,b,c).

Adapt LCSs to extract association rules online. Under the fourth objective of this thesis, we designed Fuzzy-UCS, preparing the system to deal with streams of labeled data with vagueness and uncertainty. This resulted in a powerful tool that was shown to be able to mine large data sets online, which were made available as data streams. We already argued that the online learning architecture made Fuzzy-UCS an appealing alternative to extract classification models from data streams, which is increasingly demanded in industrial applications.

In addition, there are many industrial applications which generate streams of unlabeled data (Aggarwal, 2007; Gama and Gaber, 2007). This defines an unsupervised learning problem, in which learning examples need to be processed on the fly to extract useful information. To solve these types of problems, we propose to use the same ideas of XCS, UCS, and Fuzzy-UCS and prepare LCSs to extract key information from these streams of data in form of association rules.

For this purpose, concepts of association rule mining would be incorporated to a system similar to Fuzzy-UCS, and the affected mechanisms of the system would be modified to deal with the new representation. Note that this future work line also puts emphasis on the crossbreeding of ideas to solve a new challenging problem in machine learning. That is, we propose to define an unsupervised learning technique by combining the best practices in LCSs, GAs, fuzzy logic, and association rule mining.

Recently, the first steps toward this direction have been taken. A Michigan-style LCS for association rule mining, which incorporates many of the mechanisms discussed in this thesis, was designed in (Orriols-Puig et al., 2008f). Later, another version of the same system, which included the fuzzy representation of Fuzzy-UCS, was used to model the consumer behavior in a web-based trust model (Orriols-Puig et al., 2008h,g). New applications as well as more analyses of the system behavior will be conducted as further work.

After all the conclusions and future work lines provided through all the chapters of this thesis and collected in the present and previous sections, little extra motivation can be given to highlight the many future lines of this work. To conclude with the thesis, I would only recall to the reader the innovation and creativity capabilities of evolution, which by means of basic genetic information (building blocks) mixing, random changes, and selection-of-the-fittest principle, has been able to provide accurate, adapted, and diverse solutions to life. Therefore, the power of competent GAs, LCSs, and GBML systems encourages their application to solve increasingly challenging problems that require a dose of innovation and creativity. Without GAs, many real-world problems would not have been solved. Surely, GAs and GBML have a lot to contribute in the following decades.

Appendix A

Description of the Artificial Problems

This appendix describes the artificial problems used along the chapters of this thesis. For each problem, we provide a concise description, specify and give a particular example of the best action map $[B]$ (Bernadó-Mansilla and Garrell, 2003) and the complete action map $[O]$ (Wilson, 1995; Kovacs and Kerber, 2001) of the problem, and describe the dimensions along which the problem difficulty can be moved.

A.1 Parity

The parity is a problem that has widely been used as a benchmark in LCS since it was originally introduced by Kovacs (2001) to show the dependence of XCS's performance on the optimal population size. The problem is defined as follows. Given a binary string of length ℓ, where there are k relevant bits ($0 < k \leq \ell$), the output is the number of one-valued bits in the k relevant bits modulo two. The complexity of the problem can be moved along the building block length, which is controlled with the parameter k. That is, larger values of k poses more complexity to the learner, which needs to discover larger building blocks, and so, larger populations, without

Table A.1: Best action map (first and second columns) and complete action map (all columns) of the parity problem with $\ell = k = 4$.

0000:0	1000:1	0000:1	1000:0
0001:1	1001:0	0001:0	1001:1
0010:1	1010:0	0010:0	1010:1
0011:0	1011:1	0011:1	1011:0
0100:1	1100:0	0100:0	1100:1
0101:0	1101:1	0101:1	1101:0
0110:0	1110:1	0110:1	1110:0
0111:1	1111:0	0111:0	1111:1

Table A.2: Best action map (first column) and complete action map (all columns) of the decoder problem with $\ell = k = 4$.

0000:0	###:0	#1##:0	##1#:0	###1:0
0001:1	###:1	#1##:1	##1#:1	###0:1
0010:2	###:2	#1##:2	##0#:2	###1:2
0011:3	###:3	#1##:3	##0#:3	###0:3
0100:4	###:4	#0##:4	##1#:4	###1:4
0101:5	###:5	#0##:5	##1#:5	###0:5
0110:6	###:6	#0##:6	##0#:6	###1:6
0111:7	###:7	#0##:7	##0#:7	###0:7
1000:8	###:8	#1##:8	##1#:8	###1:8
...	...	...	...	...
1111:15	###:15	#0##:15	##0#:15	###0:15

fitness guidance.

The best action map size consists of 2^ℓ rules, each one with all the k relevant bits specified and predicting the correct class. The complete action map doubles the best action map, as it also maintains specific rules predicting the wrong class; thence, the size of the complete action map is $|[O]| = 2^{\ell+1}$. Table A.1 shows the best action map and the correct action map for a parity problem with $\ell = 4$ and $k = 4$.

A.2 Decoder

The decoder problem is an artificial problem with binary inputs and multiple classes. Given an input of length ℓ, where there are k relevant bits ($0 < k \leq \ell$), the output is determined by the decimal value of k relevant bits. Therefore, the k relevant bits form a building block. The number of classes increases exponentially with the condition length, that is, $num_{classes} = 2^\ell$. Thus, k controls three dimensions of complexity: the number of classes, the length of the building blocks of the problem, and the size of the optimal population.

The best action map is formed by 2^ℓ classifiers, each one with the k relevant variables specified and the class set to the decimal value of the k relevant bits. The complete action map adds ℓ consistently incorrect rules per each consistently correct rule of the best action map; thus, $|[O]| = 2^\ell \cdot (\ell + 1)$. Table A.2 shows the best action map and the correct action map for a decoder problem with $\ell = 4$ and $k = 4$. Note that whilst the best action map contains classifiers with all k bits specified, the correct action map contains consistently incorrect classifiers that have all bits general but one.

A.3 Position

Position is an imbalanced multi-class problem defined as follows. Given a binary-input instance of length ℓ, the output is the position of the left-most one-valued bit. The best action map

Table A.3: Best action map (first column) and complete action map (all columns) of position with ℓ=6.

000000:0	1#####:0	#1####:0	##1###:0	###1##:0	####1#:0	#####1:0
000001:1	1#####:1	#1####:1	##1###:1	###1##:1	####1#:1	#####0:1
00001#:2	1#####:2	#1####:2	##1###:2	###1##:2	####1#:2	
0001##:3	1#####:3	#1####:3	##1###:3	###0##:3		
001###:4	1#####:4	#1####:4	##0###:4			
01####:5	1#####:5	#0####:5				
1#####:6	0#####:6					

Table A.4: Best action map (first column) and complete action map (all columns) of the multiplexer problem with $\ell = 6$.

000###:0	000###:1
001###:1	001###:0
01#0##:0	01#0##:1
01#1##:1	01#1##:0
10##0#:0	10##0#:1
10##1#:1	10##1#:0
11###0:0	11###0:1
11###1:1	11###1:0

consists of $\ell + 1$ rules with different levels of generalization. The complete action map needs to maintain $\frac{\ell^2+3\ell}{2}$ consistently incorrect rules; thence, the size of the complete action map is $|[O]| = \ell + 1 + \frac{\ell^2+3\ell}{2}$ Table A.3 shows the best action map and the complete action map for position with $\ell = 6$. Notice that k controls four dimensions of complexity: the number of classes, the length of the building block, the size of the optimal population, and the imbalance ratio between the most general and the most specific optimal classifier.

A.4 Multiplexer

The multiplexer problem is one of the most used benchmarks in accuracy-based learning classifier systems (Wilson, 1995). The multiplexer is defined for binary strings of size ℓ, where the first $\log_2 \ell$ bits are the address bits and the remaining bits are the position bits. Then, the output is the value of the position bit referred by the decimal value of the address bits. Note that k controls two dimensions of problem difficulty: the length of the building block and the size of the optimal population.

The best action map consists of $2^{log_2 \ell+1}$ classifiers with all the address bits and the corresponding position bit specified and all the other bits set to '#'. The complete action adds $2^{log_2 \ell+1}$ classifiers with the same conditions as those in the best action map, but the opposite class; therefore, $|[O]| = 2^{log_2 \ell+2}$. Table A.4 shows an example of the best action map and the complete action map for the multiplexer problem with $\ell = 6$.

A.4.1 Imbalanced Multiplexer

The imbalanced multiplexer was introduced in (Orriols-Puig and Bernadó-Mansilla, 2006a) to analyze the effects of under-sampled classes in XCS. Departing from the original multiplexer problem, the imbalanced multiplexer under-samples one of the classes—labeled as the minority class—so that the ratio of the number of instances of class 0 to the number of instances of class 1 sampled to the system equals to the parameter ir, which is specified by the user. The best and the complete action map are the same than those of the original multiplexer problem. Therefore, the system has to generalize the same optimal population regardless of the fact that the instances of one of the classes are under-sampled. Note that the imbalanced multiplexer enables moving the complexity of the problem along three dimensions: the length of the building block, the size of the optimal population, and the imbalance ratio.

A.4.2 Multiplexer with Alternating Noise

The multiplexer with alternating noise was firstly used in (Butz, 2006) to show that XCS with tournament selection is able to handle data sets with inconsistent data. The problem modifies the multiplexer by adding alternating noise. That is, when a new input instance corresponding to the multiplexer problem is sampled, its action is flipped with probability P_x. Again, the best and the complete action map are the same than those of the original multiplexer problem. Note that the multiplexer with alternating noise permits moving the complexity of the problem along three dimensions: the length of the building block, the size of the optimal population, and the proportion of noise in the class of the learning instances.

Appendix B

Statistical Tests

This appendix describes the statistical tests employed along this thesis. We first briefly motivate the use of non-parametric statistical tests for multiple and pairwise comparisons. Then, we describe the methodology used to statistically analyze the results in the experimental comparisons of this thesis. This methodology contemplates comparisons among multiple learners and comparisons between pairs of learners. The procedures followed and the particular tests used in each of the two types of comparisons are explained in detail.

B.1 Statistical Tests for Contrasting Hypotheses

The strong research on machine learning has resulted in the design of several learning algorithms whose behavior is typically compared with existing learning systems on a collection of data sets. For this purpose, the machine learning community has agreed in some common procedures to set up the experiments. Currently, most of the published papers use validation procedures such as k-fold cross validation (Dietterich, 1998) to obtain reliable estimates of the metric employed to assess the quality of the learners. In turn, this quality can be measured with different metrics; for example, in this thesis, we have used the test accuracy, the product of TP rate and TN rate, and the size of the models as some of these indicators. This procedure yields large tables of results in which, for each data set and method, we have n measures of performance ($n \geq 1$) that have been obtained by applying the method in each validation set and repeating this procedure for multiple seeds in case that either the validation process or the learning algorithm is stochastic.

The purpose of this chapter is to provide detail on the statistical tests employed to analyze these tables of results. The underlying idea is to apply statistical methods to contrast our initial hypotheses. Some examples of these hypotheses could be

- "method A outperforms method B", for pairwise comparisons, or
- "method A is the best of the comparison, surpassing the results obtained with methods B, C, and D", for multiple comparisons.

All the statistical methodology presented herein follows the recommendations by Demšar (2006), who emphasizes the importance of using non-parametric statistical tests for the types of comparisons usually performed in machine learning. The reason for this is clearly stated by the author;

in general, parametric tests require that the input data—in our case, the tables of results, which consist of the performance of the compared methods—satisfy strong conditions, and the tests to check these conditions need large amounts of data—that is, a large number of data sets—to be effective (for further details, please see (Demšar, 2006)). This is not usually the case in machine learning. For this reason, all the statistical analyses conducted in this thesis have been based on non-parametric tests.

In the following sections, we describe the methodology used to compare (1) pairs of learners (section B.2), and (2) multiple learners (section B.3). All these tests are computed on tables of results in which, for each data set and learner, a single performance measure is provided; in case of multiple runs, the average performance is supplied. For each statistical test, we provide a general description of which type of null hypotheses the statistical tests check and carefully describe the process followed by the test, illustrating the procedure with an example.

B.2 Comparisons of Two Learning Systems

When two algorithms are compared, we aim at contrasting the null hypothesis of whether *"algorithm A significantly outperforms algorithm B on a collection of problems"*. Following the recommendations by Demšar (2006), in our statistical comparisons we avoided using the *paired t-test* parametric test (Sheskin, 2000), probably the most used test for pairwise comparisons in machine learning in the last decade. Although this common use, the main drawback of the paired t-test is that it requires three conditions on the data: commensurability of the data, normal distribution of the differences between the two random variables, and lack of outliers. Since these three conditions are hard to satisfy in the typical machine learning comparisons, our statistical analysis used the non-parametric counterpart of the paired t-test, that is, the *Wilcoxon signed-ranks test* (Wilcoxon, 1945). The Wilcoxon signed-ranks test can be reliably applied to the average of the performance measure computed for a machine learning technique and imposes no additional restrictions to the data. As proceeds, this test is explained in more detail.

B.2.1 The Wilcoxon Signed-Ranks Test

The Wilcoxon signed-ranks test ranks the differences in the performance of two classifiers for each data set, ignoring the signs, and compares the ranks for the positive and the negative differences. As proceeds, we explicate the procedure to compute the statistic for this test. Besides, we exemplify the procedure by applying the statistical test on the results of table B.1, in which the performance of two methods, let us say method M1 and method M2, are compared on a collection of 20 data sets. For each method and data set, the table provides an average value of performance.

The first step of the test is to compute the differences of the performance measures obtained by each method in each data set. These differences are calculated in the fourth column of table B.1. Then, the absolute values of these differences are ranked, considering that the smallest difference holds the first position of the ranking, and the largest difference gets the last position of the ranking. In case of ties, the average rank is assigned to all the elements that have the same performance.

Table B.1: Comparison of the performance of methods M1 and M2 (second and third column). The fourth column provides the performance difference, and the fifth column supplies the rank of the differences.

	M1	**M2**	**difference**	**rank**
ann	97.39	98.61	1.22	11
aut	67.42	69.32	1.90	14
bal	84.40	83.40	-1.00	9
bpa	59.42	58.93	-0.49	7
cmc	49.67	49.42	-0.25	5
col	82.46	78.50	-3.96	18
gls	57.21	57.43	0.22	4
h-c	82.62	82.05	-0.57	8
h-s	80.78	78.11	-2.67	16
irs	95.47	93.73	-1.74	12
pim	74.11	74.32	0.21	3
son	73.71	71.66	-2.05	15
tao	83.02	87.53	4.51	20
thy	89.49	91.25	1.76	13
veh	65.35	65.34	-0.01	1
wbcd	95.73	95.29	-0.44	6
wdbc	94.61	94.51	-0.10	2
wne	94.86	91.82	-3.04	17
wpbc	76.05	71.69	-4.36	19
zoo	94.78	95.90	1.12	10

Thereafter, the method computes R^+ as the sum of ranks for the data sets on which M2 outperformed M1, and R^- as the sum of ranks for the data sets on which M1 outperformed M2. Ranks for which the difference is 0 are split evenly between R^+ and R^-. If there is an odd number of them, one is ignored. In the example, we obtain $R^+ = 75$ and $R^- = 135$.

Then, we assign to T the smaller value between R^+ and R^-, that is, $T = min(R^+, R^-)$. With T, we can compute the z statistic as

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}, \tag{B.1}$$

where z is distributed normally and N is the number of data sets of the comparison. With $\alpha = 0.05$, the null hypothesis can be rejected if z is smaller than -1.96. The exact *p-value* can be extracted from the table of the normal distribution.

Let us apply this final step to our example. Replacing $T = 75$ and $N = 20$ into equation B.1, we obtain that $z = -1.11$. As $z < -1.96$, we cannot reject the hypothesis that both learners perform the same, on average, with $\alpha = 0.05$. Besides, by consulting the table of the normal distribution, we can compute the exact p-value by checking the probability whose value approaches the z statistic the most; in this case, we obtained $p = 0.26$.

Table B.2: Comparison of the performance of methods M1, M2, and M3. For each method and data set, the average rank is supplied in parentheses. The last column provides the rank of each learning algorithm for each data set.

	M1	**M2**	**M3**
ann	98.85 (1)	97.39 (3)	98.61 (2)
aut	74.42 (1)	67.42 (3)	69.32 (2)
bal	88.65 (1)	84.40 (2)	83.40 (3)
bpa	59.82 (1)	59.42 (2)	58.93 (3)
cmc	51.72 (1)	49.67 (2)	49.42 (3)
col	85.01 (1)	82.46 (2)	78.50 (3)
gls	60.65 (1)	57.21 (3)	57.43 (2)
h-c	84.39 (1)	82.62 (2)	82.05 (3)
h-s	81.33 (1)	80.78 (2)	78.11 (3)
irs	95.67 (1)	95.47 (2)	93.73 (3)
pim	74.88 (1)	74.11 (3)	74.32 (2)
son	80.78 (1)	73.71 (2)	71.66 (3)
tao	81.71 (3)	83.02 (2)	87.53 (1)
thy	88.18 (3)	89.49 (2)	91.25 (1)
veh	67.68 (1)	65.35 (2)	65.34 (3)
wbcd	96.01 (1)	95.73 (2)	95.29 (3)
wdbc	95.20 (1)	94.61 (2)	94.51 (3)
wne	94.12 (2)	94.86 (1)	91.82 (3)
wpbc	76.06 (1)	76.05 (2)	71.69 (3)
zoo	96.50 (1)	94.78 (3)	95.90 (2)
Avg	1.25	2.20	2.55

B.3 Comparisons of Multiple Classifiers

In a multiple classifier comparison, the machine learning practitioner may think of using pairwise comparisons repeatedly, so that all possible pairs of algorithms are compared against each other. Although this has been a common practice in machine learning, statisticians have warned that this procedure causes a certain proportion of the null hypotheses to be rejected due to random chance (Sheskin, 2000; Demšar, 2006). In fact, testing multiple hypotheses is a well-known problem is statistics. Along this thesis, we have followed the methodology proposed by Demšar (2006), which is based on non-parametric statistical tests. This methodology proposes to first apply a multiple-comparison test to analyze whether all the algorithms performed the same on average. If this is the case, no further actions can be taken. Otherwise, different post-hoc tests can be applied depending on the null hypothesis to be tested.

As proceeds, we describe the three multiple-comparison tests used in this thesis. We first describe the Friedman's tests (Friedman, 1937, 1940), which contrasts the null hypothesis of whether all the algorithms perform the same on average. Then, we explain the Nemenyi test (Nemenyi, 1963), a post-hoc test that compares all learning algorithms with each other. Finally, we describe the Bonferroni-Dunn procedure (Dunn, 1961), which compares all the learning algorithms with another one that is selected as the control classifier. To exemplify the procedures, we apply each test on the results of the comparison of three learners provided in table B.2.

B.3.1 Friedman's Test

The Friedman's test (Friedman, 1937, 1940) is a non-parametric multiple-comparison test that is used to detect significant differences across multiple methods. That is, the procedure checks the null hypothesis of "whether all algorithms perform the same on average". The procedure is similar to its parametric counter part, the ANOVA test (Fisher, 1959). The key difference between them is that the Friedman's test is based on the ranks of the algorithms and does not require further assumptions of normality and sphericity of the data (Demšar, 2006). As proceeds, the statistical procedure is detailed and applied to the example in table B.2.

The procedure starts ranking the algorithms for each data set (see the ranks in parentheses in table B.2). In case of ties, the average rank is assigned to each learner. Then, the procedure computes the average rank R_i of each algorithm i, which is provided in the last row of table B.2. Next, the Friedman statistic is computed as

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_i R_i^2 - \frac{k(k+1)^2}{4} \right] \tag{B.2}$$

where N is the number of data sets, and k is the number of learning algorithms in the comparison. The Friedman statistic is distributed according to the χ^2 distribution with $k-1$ degrees of freedom.

Let us now calculate the Friedman statistic for the example in table B.2. Replacing $N = 20$ and $k = 3$ in equation B.2 we obtain that

$$\chi_F^2 = \frac{12 \cdot 20}{3 \cdot 4} \left[(1.25^2 + 2.20^2 + 2.55^2) - \frac{3 \cdot 4^2}{4} \right] = 18.1 \tag{B.3}$$

With three algorithms, the statistic behaves as the χ^2 distribution with 2 degrees of freedom. Hence, the critical value of $\chi^2(2)$ for $\alpha = 0.05$ is 10.60. As the computed Friedman statistic is greater than 10.60, we can reject the null hypothesis that all the learners perform the same on average. Besides, we can use the same table to provide the p-value by checking the probability whose value is the closest to the obtained Friedman statistic; in this example, p=0.0001.

When the Friedman procedure rejects the null hypothesis, a post-hoc test is applied to detect further differences. The next subsections explicate two of these procedures: the Nemenyi test and the Bonferroni-Dunn test.

B.3.2 Post-hoc Nemenyi Test

The post-hoc Nemenyi test aims at comparing all classifiers with each other. The method is based on the critical distance among learners, that is, the minimum distance between two methods to consider that they are statistically different. The procedure is detailed as follows.

Table B.3: Critical values for the two-tailed Nemenyi test.

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

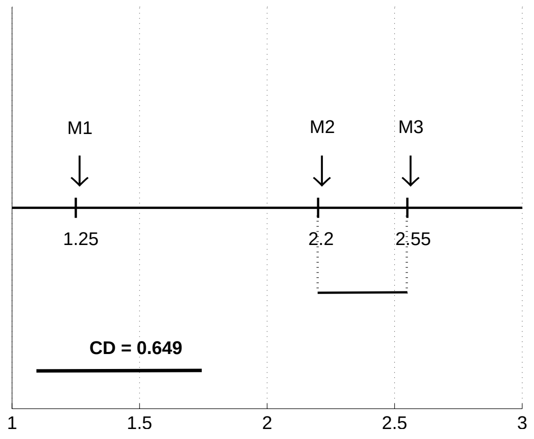


Figure B.1: Comparison of the performance of all classifiers against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.10$) are connected.

The Nemenyi test considers that the performance of two classifiers is different if the distance between their ranks is larger than the critical distance CD computed as

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{B.4}$$

where N and k are the number of learners and the number of data sets respectively, and q_α is the critical value based on the Studentized range statistic (Sheskin, 2000). Table B.3 provides the critical values for the Nemenyi test for $\alpha = \{0.05, 0.01\}$ and from $k = 2$ to $k = 10$.

The critical distance for the example in table B.2 is calculated as follows. Recognizing that $k = 3$, we can extract, from table B.3, that $q_{0.10} = 2.052$ at $\alpha = 0.10$. Thence, the critical distance is

$$CD_{\alpha=0.10} = 2.052\sqrt{\frac{3 \cdot 4}{6 \cdot 20}} = 0.649 \tag{B.5}$$

Therefore, any pair of algorithms whose rank differs by more than 0.649 perform significantly different. These results are illustrated in figure B.1 in which each learner is depicted according to its rank, and all the algorithms that perform equivalently are connected with a line.

Table B.4: Critical values for the two-tailed Bonferroni-Dunn test.

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.241	2.394	2.498	2.576	2.638	2.690	2.724	2.773
$q_{0.10}$	1.645	1.960	2.128	2.241	2.326	2.394	2.450	2.498	2.539

B.3.3 Post-hoc Bonferroni-Dunn Test

If we want to compare all learning systems with respect to a control learner, we can use the Bonferroni-Dunn (Dunn, 1961) test instead of the Nemenyi test. As proceeds, this statistical procedure is explained in detail.

The Bonferroni-Dunn test provides a general procedure to control the family-wise error in multiple hypotheses tests by dividing α by the number of comparisons that have to be performed; in our case, we perform $k-1$ comparisons since each learner is compared with the control algorithm. The statistical procedure can be computed in two different ways. Firstly, the statistic for comparing the ith and the jth classifiers can be calculated as

$$z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}} \tag{B.6}$$

where R_i and R_j are the ranks of the ith and the jth learners, k is the number of learning systems in the comparison, and N is the number of data sets. Once computed, the z value is used to find the corresponding probability from the table of normal distribution, which can be compared with the desired α. As mentioned, the significance level needs to be adjusted as $\alpha/(k-1)$.

The second equivalent procedure to compute the Bonferroni-Dunn test is by using the concept of critical distance, as done by the Nemenyi test. The critical distance is computed as indicated in equation B.4, where the values of q_α are calculated from the Studentized range statistic, but with the difference that $\alpha/(k-1)$ instead of α is considered to obtain these values. Table B.4 provides the critical values for the Bonferroni-Dunn test for $\alpha = \{0.05, 0.01\}$ and from $k = 2$ to $k = 10$. The advantage of applying this second procedure to compute the Bonferroni-Dunn test is that the results can be visually illustrated, as done for the Bonferroni-Dunn test.

Let us now compute the Bonferroni-Dunn test for the example in table B.2. The first step is to identify the control learner with which the classifiers will be compared. Let us assume that we want to compare the different classifiers with the best rank method, that is, M1. Then, having that $k = 3$, table B.3 indicates that $q_{0.10} = 1.960$ at $\alpha = 0.10$. Thence, the critical distance is

$$CD = 1.960\sqrt{\frac{3 \cdot 4}{6 \cdot 20}} = 0.619 \tag{B.7}$$

Then, we compare the differences of ranks between each classifier and the control learner. The difference between the ranks of M1 and M2 is 0.95 which is greater than 0.619; therefore, M2 significantly degrades the results of M1. Similarly, the difference between the ranks of M1 and M3 is 1.30, which in turn is greater than 0.619; thence, M1 also significantly outperforms M3 according to a Bonferroni-Dunn test at $\alpha = 0.10$. The same graphical representation as the one done for the Nemenyi test can be used here. Nonetheless, note that, in this case, M1 significantly outperforms all the other methods, so no graphic representation is necessary.

B.4 Summary

This appendix has described the statistical tests used along the thesis. We first briefly discussed the robustness of non-parametric tests with respect to parametric tests and draw the methodology used in this thesis to compare pairs of classifiers and multiple (more than two) learning systems. Then, each of these two types of comparisons got a different section where the particular statistical methods used in our analyses were described in detail, providing a detailed example of use for each one.

Appendix C

Full Results of the Comparison of the Re-sampling Techniques

Section 7.5 analyzed whether the application of re-sampling methods improved the accuracy of the models extracted by XCS, UCS, C4.5, SMO, and IBk on imbalanced domains. For compactness, the analysis only gathered the statistical analysis and extracted conclusions from it. The purpose of this appendix is to provide the full results of the comparison, which involved the following four re-sampling techniques: (1) random over-sampling, (2) under-sampling based on Tomek links, (3) SMOTE, and (4) cSMOTE. As proceeds, we describe and provide the tables of results.

Description of the Tables of Results

Tables C.1, C.2, C.3, C.4, and C.5 supply the average of the product of the TP rate and the TN rate obtained in each data set by C4.5, SMO, IBk, XCS, and UCS respectively. Moreover, to let a more detailed analysis of each problem, we also provide pair-wise comparisons for each particular combination of learner and re-sampling technique per data set. The • and ◦ symbols denote a significant degradation/improvement of the method in the corresponding data set with respect to the same data set but with another re-sampling method (or without re-sampling). We acknowledge in advance that pair-wise comparisons may be taken with a grain of salt; nevertheless, they also help draw good insights about the performance of each learner. Finally, the last four rows of the table summarize (i) the average performance, (ii) the average rank, (iii) the position of each learner in the ranking, and (iv) the number of times that the results obtained with the learner are surpassed/degraded by another learner.

Table C.1: Comparison of the performance, measured as the product of TP rate and TN rate, achieved by **C4.5** with the original and re-sampled data sets. For each method and data set, the • and ◦ symbols indicate that the method is statistically inferior/superior than another of the learners according to a Wilcoxon signed-ranks test at $\alpha = 0.05$. *Avg* provides the performance average of each method over the 25 data sets. Rows *Rank* and *Pos* show the average rank of each learning algorithm and its position in the ranking respectively. The last row provides *Inf/Sup*, where *Inf* is the number of times that the learner has been surpassed by another one, and *Sup* is the number of times that the method has outperformed another one.

	Original	**Ovs**	**UnsTL**	**SMOTE**	**cSMOTE**
bald1	0.00 ± 0.00 •	19.91 ± 37.27 ◦◦◦◦	0.00 ± 0.00 •	0.00 ± 0.00 •	1.90 ± 6.32 •
bald2	69.30 ± 6.83 •	67.57 ± 6.85 ••	68.03 ± 6.40 •	72.73 ± 8.28 ◦	75.35 ± 6.88 ◦◦◦
bald3	71.20 ± 6.04 •◦	68.31 ± 5.55 •	65.78 ± 4.91 •••	72.82 ± 4.21 ◦	77.75 ± 6.87 ◦◦◦
bpa	33.08 ± 14.09	36.81 ± 11.78	31.67 ± 17.31	33.31 ± 7.49	29.56 ± 15.74
glsd1	79.50 ± 42.16	89.50 ± 31.62	89.81 ± 0.00 •	99.50 ± 0.00 ◦◦	59.00 ± 51.64 •
glsd2	34.50 ± 47.43 •	82.50 ± 33.75 ◦	58.50 ± 48.30	63.50 ± 47.43	68.00 ± 42.16
glsd3	28.97 ± 42.16 •	46.45 ± 47.14	45.96 ± 35.36	64.95 ± 42.16 ◦	24.22 ± 35.36
glsd4	73.55 ± 32.63	80.70 ± 25.40	71.73 ± 19.95 •	84.93 ± 19.33 ◦	74.29 ± 32.63
glsd5	66.52 ± 16.77 ◦	70.29 ± 17.10 ◦	52.12 ± 15.06 ••••	64.77 ± 16.56 ◦	66.68 ± 15.43 ◦
glsd6	52.54 ± 15.13	53.80 ± 9.75	49.78 ± 15.69	54.54 ± 24.12	46.34 ± 25.53
h-s	63.33 ± 13.29	58.06 ± 9.98	61.17 ± 15.84	59.44 ± 14.80	56.67 ± 13.72
pim	43.87 ± 13.27 •	54.68 ± 7.25 ◦◦	50.50 ± 12.19	52.38 ± 8.98	46.76 ± 7.87 •
tao	90.98 ± 2.14 •	90.98 ± 2.14 •	91.10 ± 1.30	92.71 ± 1.53 ◦◦	91.86 ± 3.96
thyd1	87.61 ± 16.10	84.63 ± 17.21	87.31 ± 14.05	82.09 ± 17.21	76.53 ± 17.21
thyd2	93.24 ± 12.45 ◦	91.94 ± 12.45	85.32 ± 13.61 •	91.11 ± 13.61	88.06 ± 16.87
thyd3	87.65 ± 10.34	88.13 ± 8.08	86.06 ± 7.81	83.25 ± 11.99	84.79 ± 8.05
wavd1	67.79 ± 4.06 •◦	67.99 ± 2.93 ◦	70.25 ± 3.36 ◦◦	70.75 ± 4.02 ◦	64.04 ± 3.50 ••••
wavd2	62.54 ± 3.89	65.05 ± 3.62 ◦	64.02 ± 3.19	64.41 ± 2.91 ◦	61.33 ± 3.56 ••
wavd3	68.60 ± 2.38 •◦	69.35 ± 3.35 •◦	70.54 ± 2.18 ◦◦	71.45 ± 3.50 ◦◦	65.66 ± 3.54 ••••
wbcd	89.12 ± 3.42	89.63 ± 4.21	90.70 ± 3.29	91.94 ± 2.18	91.94 ± 5.17
wdbc	88.79 ± 5.09	86.51 ± 6.88	85.95 ± 5.01	87.67 ± 3.30	86.53 ± 5.04
wined1	85.15 ± 16.63	89.62 ± 16.63	79.92 ± 16.36	89.92 ± 16.36	89.69 ± 13.98
wined2	91.81 ± 8.05	89.38 ± 8.05	87.71 ± 8.05 •	91.74 ± 8.05 ◦	88.47 ± 8.78
wined3	87.62 ± 11.70	84.46 ± 11.68	84.03 ± 9.64	84.61 ± 6.90	83.36 ± 11.76
wpbc	33.55 ± 12.87	33.30 ± 21.92	30.19 ± 13.29	38.36 ± 21.66	30.66 ± 28.36
Avg	66.03	70.38	66.33	70.52	65.18
Rank	3.00	2.60	3.68	2.14	3.58
Pos	3	2	5	1	4
Inf/Sup	9/5	5/11	13/4	1/14	13/7

Table C.2: Comparison of the performance, measured as the product of TP rate and TN rate, achieved by **SMO** with the original and re-sampled data sets. For each method and data set, the • and ◦ symbols indicate that the method is statistically inferior/superior than another of the learners according to a Wilcoxon signed-ranks test at $\alpha = 0.05$. *Avg* provides the performance average of each method over the 25 data sets. Rows *Rank* and *Pos* show the average rank of each learning algorithm and its position in the ranking respectively. The last row provides *Inf/Sup*, where *Inf* is the number of times that the learner has been surpassed by another one, and *Sup* is the number of times that the method has outperformed another one.

	Original	**Ovs**	**UnsTL**	**SMOTE**	**cSMOTE**
bald1	0.00 ± 0.00 •	16.38 ± 13.78 ◦◦◦◦	0.00 ± 0.00 •	0.00 ± 0.00 •	0.00 ± 0.00 •
bald2	84.03 ± 7.30 ◦◦	84.28 ± 7.11 ◦◦	85.09 ± 6.50 ◦◦	76.94 ± 8.26 •••	77.88 ± 9.00 •••
bald3	85.81 ± 8.40 •◦	85.43 ± 9.16 •◦	85.24 ± 6.11 •◦	78.06 ± 9.85 ••••	90.13 ± 7.76 ◦◦◦◦
bpa	0.00 ± 0.00 •••	36.33 ± 6.75 ◦◦◦◦	11.99 ± 4.16 ••◦◦	0.00 ± 0.00 •••	28.18 ± 11.16 •◦◦◦
glsd1	0.00 ± 0.00 ••	87.83 ± 7.60 ◦◦◦	71.71 ± 24.59 ◦◦◦	10.00 ± 31.62 ••	0.00 ± 0.00 ••
glsd2	15.00 ± 33.75 •••	82.50 ± 29.74 ◦◦	67.00 ± 36.61 •◦◦	74.50 ± 39.61 ◦◦◦	15.00 ± 33.75 •••
glsd3	0.00 ± 0.00 ••	30.36 ± 11.39 ◦◦◦	21.03 ± 16.84 ◦◦◦	0.00 ± 0.00 ••	0.00 ± 0.00 ••
glsd4	80.03 ± 24.33	85.09 ± 17.44 ◦	82.03 ± 16.81 ••	85.61 ± 17.83 ◦	82.99 ± 19.30
glsd5	9.50 ± 9.42 ••••	44.67 ± 16.83 ◦◦	38.33 ± 13.33 •◦	42.57 ± 14.42 ◦	28.65 ± 22.27 ◦
glsd6	0.00 ± 0.00 ••••	26.89 ± 13.24 ◦◦	27.36 ± 11.27 ◦◦	17.71 ± 8.40 ••◦	22.28 ± 14.58 ◦
h-s	68.83 ± 8.87 ◦	66.89 ± 9.24	67.61 ± 7.32	65.83 ± 12.08	63.78 ± 8.70 •
pim	48.31 ± 5.60 •••	55.75 ± 7.12 ◦	56.25 ± 6.88 ◦◦	49.79 ± 8.14 •	53.44 ± 8.05 ◦
tao	70.59 ± 6.45 ◦◦	70.59 ± 6.45 ◦◦	70.49 ± 6.15 ◦	62.44 ± 5.91 ••••	69.28 ± 6.92 ••◦
thyd1	76.67 ± 22.50 •	90.00 ± 16.10 ◦◦	80.00 ± 23.31	80.00 ± 23.31	76.67 ± 22.50 •
thyd2	54.17 ± 24.92 ••••	98.33 ± 2.68 ◦◦	93.61 ± 12.29 ◦	96.39 ± 7.86 ◦◦	80.83 ± 18.02 ••◦
thyd3	33.81 ± 21.35 •••	52.38 ± 19.76 ◦◦◦	41.67 ± 21.68 ••	62.13 ± 18.65 ◦◦◦	39.76 ± 24.49 ••◦
wavd1	78.68 ± 4.27 ••	80.98 ± 3.00 ◦◦	80.81 ± 2.90 ◦◦	80.35 ± 2.37	78.84 ± 3.73 ••
wavd2	72.30 ± 2.71 ••	75.15 ± 2.18 ◦◦	74.67 ± 1.69 ◦	74.40 ± 2.56	73.75 ± 1.80 •
wavd3	79.57 ± 2.04	81.09 ± 1.35 ◦	80.84 ± 1.49	80.55 ± 1.57 •	79.75 ± 2.58
wbcd	92.70 ± 5.32 •	93.70 ± 5.06	93.31 ± 5.64 •	95.30 ± 4.68 ◦◦	93.12 ± 6.19
wdbc	94.28 ± 3.28	93.64 ± 4.66	93.60 ± 3.04	93.63 ± 4.82	92.80 ± 4.71
wined1	98.46 ± 3.24	98.46 ± 3.24	96.15 ± 4.05	98.46 ± 3.24	96.46 ± 6.61
wined2	97.50 ± 5.62	97.50 ± 4.03	95.00 ± 4.30	96.67 ± 4.30	96.67 ± 5.83
wined3	97.14 ± 6.02	95.71 ± 6.90	92.94 ± 8.14	95.23 ± 6.36	94.45 ± 7.93
wpbc	9.37 ± 16.98 ••••	43.76 ± 16.91 ◦	38.92 ± 19.79 ◦	42.35 ± 18.95 ◦	43.85 ± 21.27 ◦
Avg	53.87	70.95	65.83	62.36	59.14
Rank	3.74	1.60	2.92	3.08	3.66
Pos	5	1	2	3	4
Inf/Sup	6/40	1/40	11/24	23/14	23/14

Table C.3: Comparison of the performance, measured as the product of TP rate and TN rate, achieved by **IBk** with the original and re-sampled data sets. For each method and data set, the • and ◦ symbols indicate that the method is statistically inferior/superior than another of the learners according to a Wilcoxon signed-ranks test at $\alpha = 0.05$. *Avg* provides the performance average of each method over the 25 datasets. Rows *Rank* and *Pos* show the average rank of each learning algorithm and its position in the ranking respectively. The last row provides *Inf/Sup*, where *Inf* is the number of times that the learner has been surpassed by another one, and *Sup* is the number of times that the method has outperformed another one.

	Original	**Ovs**	**UnsTL**	**SMOTE**	**cSMOTE**
bald1	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	4.06 ± 9.56	0.00 ± 0.00
bald2	81.16 ± 5.54 ◦◦	79.66 ± 5.09 ◦	73.83 ± 3.26 •••	78.60 ± 6.09 ◦	75.25 ± 6.45 •
bald3	82.11 ± 8.67 ◦◦	80.95 ± 7.66 ◦	75.25 ± 6.21 ••	78.32 ± 8.46 •	79.61 ± 10.01
bpa	32.40 ± 9.44	36.10 ± 13.80 ◦	32.68 ± 11.87	28.97 ± 8.57 •	32.35 ± 13.56
glsd1	69.32 ± 48.30	86.94 ± 31.62	65.07 ± 0.00	87.81 ± 31.62	68.28 ± 48.30
glsd2	24.13 ± 35.36 •••	79.94 ± 33.75 ◦	68.80 ± 42.16 ◦	84.62 ± 31.62 ◦	72.75 ± 42.49
glsd3	0.00 ± 0.00 ••	38.64 ± 43.78 ◦	37.01 ± 35.36 ◦	44.70 ± 33.33	28.03 ± 34.96
glsd4	77.07 ± 24.98	76.23 ± 24.91	76.84 ± 25.40	76.64 ± 24.91	77.92 ± 24.91
glsd5	62.26 ± 21.14 ◦	62.37 ± 15.72	58.37 ± 14.75 •	62.57 ± 15.06	62.52 ± 17.88
glsd6	61.74 ± 18.23	61.70 ± 15.76	59.84 ± 24.83	63.19 ± 12.46	60.99 ± 17.86
h-s	64.40 ± 14.65	61.52 ± 10.24	59.11 ± 11.95	60.63 ± 7.91	61.50 ± 13.03
pim	46.91 ± 4.84	50.27 ± 11.30 ◦	51.50 ± 9.82 ◦	49.65 ± 6.03 ◦	44.05 ± 11.48 •••
tao	94.25 ± 2.10 ◦◦	92.61 ± 2.00	92.61 ± 2.17 •	93.02 ± 2.29	91.92 ± 2.15 •
thyd1	76.67 ± 22.50	91.26 ± 14.05	76.09 ± 23.31	84.28 ± 23.31	81.51 ± 23.57
thyd2	77.90 ± 21.40 ••••	98.33 ± 0.00 ◦	95.88 ± 7.91 ◦	98.33 ± 0.00 ◦	93.98 ± 10.54 ◦
thyd3	81.12 ± 16.16	92.38 ± 6.55	87.31 ± 7.84	88.81 ± 8.03	87.84 ± 11.45
wavd1	72.28 ± 3.97 ◦◦	71.62 ± 2.14 ◦◦	72.34 ± 2.42 ◦◦	66.67 ± 0.77 •••	65.71 ± 2.93 •••
wavd2	67.49 ± 1.75 ◦◦◦	65.62 ± 1.79 •◦◦	66.69 ± 2.50 ◦◦	57.51 ± 0.76 ••••	61.08 ± 3.58 •••◦
wavd3	74.14 ± 2.86 ◦◦	73.71 ± 1.98 •◦◦	74.81 ± 2.70 ◦◦◦	68.32 ± 0.95 •••◦	65.53 ± 2.96 ••••
wbcd	92.72 ± 5.36	94.91 ± 2.13	93.53 ± 3.83	95.02 ± 1.32	92.34 ± 4.49
wdbc	93.47 ± 3.64	91.70 ± 3.51	93.59 ± 3.51	91.45 ± 2.70	91.21 ± 4.53
wined1	94.98 ± 8.29	96.15 ± 0.00	93.85 ± 0.00	97.69 ± 0.00	96.92 ± 0.00
wined2	97.50 ± 4.03 ◦	95.76 ± 0.00 ◦	92.35 ± 0.00 ••••	95.76 ± 0.00 ◦	96.59 ± 0.00 ◦
wined3	87.94 ± 12.53	91.43 ± 9.99	92.01 ± 7.53	95.71 ± 6.90	93.34 ± 7.38
wpbc	28.98 ± 16.49	29.36 ± 20.88 •	27.70 ± 22.66 ••	37.39 ± 19.73 ◦◦	37.10 ± 21.96 ◦
Avg	65.64	71.97	68.68	71.59	68.73
Rank	2.98	2.56	3.52	2.44	3.50
Pos	3	2	5	1	4
Inf/Sup	9/15	3/14	13/11	12/8	15/4

Table C.4: Comparison of the performance, measured as the product of TP rate and TN rate, achieved by **XCS** with the original and re-sampled data sets. For each method and data set, the • and ◦ symbols indicate that the method is statistically inferior/superior than another of the learners according to a Wilcoxon signed-ranks test at $\alpha = 0.05$. *Avg* provides the performance average of each method over the 25 data sets. Rows *Rank* and *Pos* show the average rank of each learning algorithm and its position in the ranking respectively. The last row provides *Inf/Sup*, where *Inf* is the number of times that the learner has been surpassed by another one, and *Sup* is the number of times that the method has outperformed another one.

	Original	**Ovs**	**UnsTL**	**SMOTE**	**cSMOTE**
bald1	0.00 ± 0.00	1.98 ± 6.27	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
bald2	71.14 ± 5.02	70.06 ± 7.81	71.58 ± 5.03	72.79 ± 9.49	73.10 ± 6.56
bald3	69.98 ± 7.23 •••	73.95 ± 4.75	73.47 ± 6.03 ◦	72.78 ± 6.71 •◦	76.15 ± 6.58 ◦◦
bpa	47.58 ± 10.92 ◦◦	45.05 ± 12.20 ◦	38.61 ± 9.89 ◦	22.40 ± 14.39 ••••	40.69 ± 11.47 •◦
glsd1	20.00 ± 42.16 ••	69.50 ± 47.98 ◦	73.00 ± 27.01 ◦	58.50 ± 50.45	59.50 ± 51.23
glsd2	59.00 ± 45.02	59.50 ± 45.49	62.75 ± 35.72	73.50 ± 41.57	63.00 ± 45.90
glsd3	0.00 ± 0.00 •••	47.99 ± 38.80 ◦◦	42.11 ± 17.78 ◦◦	42.22 ± 40.93 ◦◦	9.74 ± 20.54 •••
glsd4	80.03 ± 24.33	87.25 ± 18.72	84.68 ± 15.68	80.59 ± 25.66	79.66 ± 24.59
glsd5	68.67 ± 18.71	66.17 ± 15.41 •	68.28 ± 19.81	73.07 ± 17.37 ◦◦	62.44 ± 16.52 •
glsd6	60.53 ± 11.21	63.48 ± 14.17	64.18 ± 12.62	64.50 ± 14.51	62.47 ± 12.64
h-s	59.89 ± 15.59	58.00 ± 11.45 •	58.61 ± 14.68	65.61 ± 15.12 ◦	60.22 ± 15.51
pim	45.85 ± 6.37 ••	50.53 ± 4.89 •◦	51.24 ± 7.64 •	55.41 ± 8.76 ◦◦◦	48.36 ± 8.97
tao	82.89 ± 5.42 •◦	83.60 ± 6.04 ◦	83.39 ± 5.91 ◦	58.01 ± 31.57 ••••	84.45 ± 6.34 ◦◦
thyd1	78.36 ± 22.01 •	87.96 ± 15.98	81.29 ± 20.83 •	91.70 ± 13.68 ◦◦	89.07 ± 16.44
thyd2	82.50 ± 24.98	95.56 ± 10.41	92.78 ± 11.43	91.94 ± 11.60	93.06 ± 12.09
thyd3	89.84 ± 11.75	87.25 ± 10.81 •	88.43 ± 11.84	93.11 ± 8.73 ◦	87.51 ± 10.38
wavd1	80.44 ± 2.97	81.91 ± 3.24 ◦	80.87 ± 3.58	80.24 ± 2.00	80.11 ± 2.97 •
wavd2	73.48 ± 2.88	76.03 ± 2.24 ◦◦	75.60 ± 1.52 ◦	71.96 ± 2.78 ••	73.84 ± 2.89 •
wavd3	81.01 ± 3.99	82.01 ± 2.05 ◦◦	81.16 ± 2.49	80.23 ± 2.19 •	79.46 ± 2.59 •
wbcd	92.31 ± 5.50	92.72 ± 6.01	92.49 ± 5.63	94.42 ± 4.41	92.70 ± 6.13
wdbc	90.27 ± 4.61	88.16 ± 6.33	90.48 ± 4.13	92.17 ± 4.95	91.08 ± 6.24
wined1	99.23 ± 2.43	95.69 ± 6.60	96.15 ± 5.44	97.69 ± 3.72	96.62 ± 8.36
wined2	99.17 ± 2.64	95.76 ± 5.96	96.67 ± 4.30	98.33 ± 3.51	97.50 ± 5.62
wined3	93.38 ± 7.15	91.86 ± 9.81	92.29 ± 6.84	94.11 ± 8.13	92.12 ± 7.74
wpbc	20.33 ± 16.38	25.84 ± 19.03	25.32 ± 17.21	31.65 ± 18.85	21.35 ± 11.96
Avg	65.83	71.11	70.62	70.28	68.57
Rank	3.60	2.86	2.94	2.42	3.18
Pos	5	2	3	1	4
Inf/Sup	12/3	4/11	2/7	12/12	8/5

Table C.5: Comparison of the performance, measured as the product of TP rate and TN rate, achieved by **UCS** with the original and re-sampled data sets. For each method and data set, the • and ◦ symbols indicate that the method is statistically inferior/superior than another of the learners according to a Wilcoxon signed-ranks test at $\alpha = 0.05$. *Avg* provides the performance average of each method over the 25 data sets. Rows *Rank* and *Pos* show the average rank of each learning algorithm and its position in the ranking respectively. The last row provides *Inf/Sup*, where *Inf* is the number of times that the learner has been surpassed by another one, and *Sup* is the number of times that the method has outperformed another one.

	Original	**Ovs**	**UnsTL**	**SMOTE**	**cSMOTE**
bald1	0.00 ± 0.00	3.23 ± 6.82	0.00 ± 0.00	3.55 ± 7.48	0.00 ± 0.00
bald2	69.75 ± 8.19	72.07 ± 6.79	72.35 ± 5.60	72.73 ± 8.03	73.22 ± 5.32
bald3	73.61 ± 6.66 •	72.18 ± 5.14 •	74.01 ± 7.05 •	73.15 ± 6.58 •	78.40 ± 6.64 ◦◦◦◦
bpa	47.59 ± 11.22	41.72 ± 10.60	48.29 ± 9.68	41.20 ± 7.60	40.74 ± 9.18
glsd1	59.00 ± 50.87	59.50 ± 51.23	72.19 ± 18.80	68.52 ± 47.33	58.52 ± 50.46
glsd2	74.00 ± 41.89 ◦	63.50 ± 46.25	77.12 ± 27.40 ◦	82.50 ± 32.68 ◦	38.50 ± 49.78 •••
glsd3	19.49 ± 25.17	33.50 ± 45.22	28.93 ± 22.94	45.24 ± 42.50	23.25 ± 24.64
glsd4	83.54 ± 19.53	87.25 ± 18.72 ◦	74.41 ± 24.70 •	82.67 ± 19.50	77.25 ± 28.64
glsd5	65.63 ± 21.46	68.54 ± 16.54	64.50 ± 14.10	62.54 ± 23.57	70.44 ± 17.42
glsd6	57.06 ± 14.20	61.64 ± 18.57	69.26 ± 21.48	62.70 ± 12.96	59.05 ± 15.39
h-s	55.00 ± 13.61	54.17 ± 16.60	55.61 ± 14.45	57.00 ± 15.95	52.83 ± 17.96
pim	47.82 ± 6.60	49.38 ± 5.11	52.45 ± 6.93 ◦	51.89 ± 8.05 ◦	46.74 ± 6.71 ••
tao	78.81 ± 7.18	80.65 ± 6.64	78.21 ± 4.27	75.72 ± 7.23	78.53 ± 7.51
thyd1	92.25 ± 13.66	88.89 ± 15.71	88.92 ± 15.51	91.88 ± 14.46	88.89 ± 15.49
thyd2	93.06 ± 12.09	91.94 ± 11.60	84.81 ± 15.61	93.33 ± 9.99	94.44 ± 10.14
thyd3	88.08 ± 14.89	84.98 ± 10.52	86.79 ± 9.80	87.95 ± 8.94	85.03 ± 10.16
wavd1	76.33 ± 2.10 ••	78.18 ± 3.10 ◦◦	78.99 ± 3.77 ◦◦	78.66 ± 3.35 ◦	75.56 ± 3.75 •••
wavd2	71.49 ± 3.83	73.08 ± 2.93 ◦	70.57 ± 2.67 •	74.46 ± 2.76 ◦◦	69.66 ± 1.60 ••
wavd3	76.60 ± 4.14 ••	80.60 ± 2.21 ◦◦	78.32 ± 2.78 ◦	79.56 ± 1.84 ◦◦	75.10 ± 2.75 •••
wbcd	94.06 ± 4.23	93.10 ± 4.97	93.10 ± 4.35	94.25 ± 4.81	94.28 ± 4.36
wdbc	89.68 ± 5.61	90.54 ± 4.08	89.84 ± 4.43	90.38 ± 7.75	87.81 ± 6.17
wined1	99.23 ± 2.43	100.00 ± 0.00 ◦◦	94.92 ± 6.49 •	96.92 ± 3.97 •	93.96 ± 9.30
wined2	91.88 ± 10.02	95.83 ± 7.08 ◦	92.56 ± 7.98 •	94.92 ± 7.07	95.83 ± 7.08
wined3	85.33 ± 9.55 ••••	94.11 ± 8.17 ◦	93.86 ± 8.52 ◦	94.71 ± 6.94 ◦	91.90 ± 5.97 ◦
wpbc	17.17 ± 21.63	21.55 ± 14.96	25.55 ± 18.72	30.70 ± 15.04	21.34 ± 13.56
Avg	68.26	69.61	69.82	71.49	66.85
Rank	3.44	2.78	2,86	2.24	3.68
Pos	4	2	3	1	5
Inf/Sup	9/1	1/10	5/6	2/8	13/5

Appendix D

Empirical Analysis of the Sensitivity of Fuzzy-UCS to Configuration Parameters

As many competitive Michigan-style LCSs, Fuzzy-UCS has several configuration parameters, which permit adjusting the behavior of the system to evolve models with maximum quality for particular problems. At a first glance, choosing a correct configuration may seem a crucial task only suitable to expert users. Nonetheless, several analyses identified the robustness of Michigan-style LCSs to the majority of configuration parameters. Actually, most of the applications of Michigan-style LCSs used the same default parameters to solve pattern recognition problems (Bernadó-Mansilla et al., 2002; Bernadó-Mansilla and Garrell, 2003; Butz, 2006; Orriols-Puig and Bernadó-Mansilla, 2008b; Dixon et al., 2002, 2004; Fu et al., 2001; Wilson, 2000). We consider that this robustness is also present in Fuzzy-UCS. For this reason, we used the same default configuration to solve the collection of real-world problems in all the experiments conducted in chapter 8.

The aim of this appendix is to empirically show the robustness of Fuzzy-UCS to configuration parameters. For this purpose, we systematically analyze the impact of the parameter settings on the quality of the final solution on a set of real-world problems; besides, we relate the results to theoretical and empirical studies of the sensitivity of LCSs—particularly XCS and UCS—to configuration parameters. It is worth highlighting that the following study does not pretend to establish guidelines to configure Fuzzy-UCS, but to intuitively show the effect of the different parameters.

The remainder of this appendix is organized as follows. Section D.1 gathers and provides a brief description all the parameters of Fuzzy-UCS. Section D.2 details the experimental methodology followed in the analysis. Section D.3 compares different configurations of Fuzzy-UCS to the default configuration used in chapter 8 and studies the impact of changing the configuration of the different parameters. Finally, section D.4 summarizes and concludes.

D.1 Configuration Parameters of Fuzzy-UCS

The configuration parameters of Fuzzy-UCS are:

1. N: Maximum population size.
2. $P_{\#}$: Generalization probability in covering.
3. ν: Fitness pressure.
4. F_0: Minimum fitness required for subsumption.
5. θ_{GA}: Threshold that controls the application period of the genetic algorithm on each niche.
6. θ_{sub}: Minimum experience required to be a candidate subsumer.
7. $\theta_{exploit}$: Minimum experience required to participate in the class inference of a new example.
8. θ_{del}: Minimum experience required to use classifier's fitness to calculate its deletion vote.
9. δ: Fraction of mean fitness below which the deletion probability of a classifier is further decreased according to the ratio of its fitness to the average fitness of the population.
10. χ: Probability of crossover.
11. μ: Probability of mutation.

D.2 Experimental Methodology

For the sake of clarity, we analyzed the effect of different parameters or groups of related parameters separately. Specifically, we examined the sensitivity of Fuzzy-UCS to:

1. Rule initialization (parameter $P_{\#}$). That is, we studied how the generalization degree in the initial population affected the quality of the models.
2. Fitness pressure (parameter ν). We analyzed to which extend the selection pressure toward highly accurate classifiers affected the learning process.
3. Genetic algorithm. We empirically showed the effect of changing a set of parameters related to the genetic algorithm: θ_{GA}, θ_{del}, and θ_{sub}.
4. Deletion (parameter δ). We examined the effects of changing the pressure toward deletion of classifiers with fitness below the average fitness.

We compared modifications on these configuration parameters with the default configuration C_P used in chapter 8, that is: N=6 400, $F_0 = 0.99$, $\nu = 10$, $\{\theta_{GA}, \theta_{del}, \theta_{sub}\} = 50$, $\theta_{exploit} = 10$, $\chi = 0.8$, $\mu = 0.6$, δ=0.1, and $P_{\#} = 0.6$, As done in chapter 8, we used the test accuracy and the size of the final rule set to evaluate the quality of the models. The results were statistically analyzed as follows. First, we applied the multiple-comparison Friedman's test (Friedman, 1937, 1940) to contrast the hypothesis that the results of all learners were equivalent on average. If significant differences were found, the post-hoc Bonferroni-Dunn test (Dunn, 1961) was used to compare a control method against the others. Moreover, pairwise comparisons were applied according to the Wilcoxon signed-ranks test (Wilcoxon, 1945).

All results provided through this appendix are averages over ten runs with different random seeds. Due to the large number of configurations run for this analysis, we restricted the data set collection to twelve of the twenty real-world problems used in chapter 8: *bal*, *bpa*, *gls*, *h-s*, *irs*, *pim*, *tao*, *thy*, *veh*, *wbcd*, *wdbc*, and *wne*.

D.3 Fuzzy-UCS's Sensitivity to Configuration Parameters

This section analyzes in detail the effect of changing the parameters related to (1) rule initialization, (2) fitness pressure, (3) genetic algorithm, and (4) deletion. Each one of these analyses gets one of the subsequent subsections.

D.3.1 Sensitivity to Rule Initialization

Population initialization was identified as a crucial aspect for the success of LCSs and evolutionary algorithms in general (Goldberg, 2002). Butz et al. (2001) theoretically derived a covering bound for XCS indicating that the initial population should be general enough to cover all the training instances and permit the genetic algorithm to take place. The theoretical study resulted in practical guidelines suggesting that the initial generalization level (i.e., the parameter $P_{\#}$ in Fuzzy-UCS) be set to a high value. The same study showed that the best results on a set of artificial problems were obtained with $P_{\#} \approx 0.6$. For this reason, as many LCS practitioners have done during the last few years, we set $P_{\#} = 0.6$ in our experiments.

Herein, we empirically analyze the effect of decreasing the generalization in the initial population of Fuzzy-UCS. For this purpose, we ran Fuzzy-UCS with the default configuration C_P, but changing $P_{\#} = 0.2$ (C_1) and $P_{\#} = 0.4$ (C_2). Tables D.1 and D.2 show the performance and the rule set size achieved by Fuzzy-UCS with the three inference schemes and the three configurations.

The multiple-comparison Friedman's test did not permit rejecting the null hypothesis that the three configurations performed equivalently on average for each inference technique of Fuzzy-UCS at 95% confidence level. Nonetheless, note that configurations C_P and C_2, that is, the configurations that use $P_{\#} = \{0.6, 0.4\}$ respectively, were the two best ranked configurations in all the inference schemes. A more detailed analysis on each particular problem permitted detecting in which problems a higher specificity on the initial population yielded more accurate models. For example, for the problems *thy*, and especially *gls*, Fuzzy-UCS obtained better results when the initial population was more specific (that is, $P_{\#}$ took a low value). On the other hand, for other problems such as *bal*, *h-s*, *pim*, *veh*, *wdbc*, and *wne*, Fuzzy-UCS presented better results with higher values of $P_{\#}$. Thus, we acknowledge that different settings of $P_{\#}$ may be used for different problems with particular characteristics and that further analysis has to be done to detect correlations between problem complexity and the setting of $P_{\#}$. Nevertheless, these results also evidence that it is safer to set $P_{\#}$ to higher values as a general rule of thumb.

Table D.1: Comparison of the test accuracy obtained with the three types of inference and the three configurations which vary $P_{\#}$. *Rank* gives the average rank of each configuration for each one of the three inference schemes. *Pos* shows the absolute position in the ranking. Frd reports the p-value obtained with the multiple-comparison Friedman test performed for each inference methodology.

	wavg			**awin**			**nfit**		
	Cp	**C1**	**C2**	**Cp**	**C1**	**C2**	**Cp**	**C1**	**C2**
bal	88.65	86.90	88.63	84.40	74.24	82.66	83.40	73.53	82.89
bpa	59.82	59.19	60.82	59.42	57.16	58.54	58.93	56.72	56.67
gls	60.65	68.77	66.10	57.21	60.53	56.60	57.43	62.76	61.81
h-s	81.33	79.26	79.89	80.78	74.81	80.63	78.11	67.07	76.32
irs	95.67	95.00	94.93	95.47	95.53	95.73	93.73	94.67	93.40
pim	74.88	73.93	74.70	74.11	73.77	74.01	74.32	71.15	72.41
tao	81.71	81.10	81.31	83.02	82.91	83.08	87.53	89.00	88.46
thy	88.18	93.93	91.28	89.49	90.66	90.09	91.25	92.75	92.23
veh	67.68	67.29	68.67	65.35	66.65	67.02	65.34	63.81	65.65
wbcd	96.01	95.35	96.11	95.73	94.61	95.78	95.29	93.97	95.58
wdbc	95.20	93.55	95.31	94.61	91.40	93.64	94.51	90.05	93.62
wne	94.12	95.34	95.04	94.86	93.51	96.06	91.82	89.74	93.40
Rank	1.83	2.42	1.75	1.83	2.50	1.67	1.83	2.25	1.92
Pos	2	3	1	2	3	1	1	3	2
Frd	0.2053			0.097			0.558		

Table D.2: Comparison of the model sizes obtained with the three types of inference and the three configurations which vary $P_{\#}$. *Rank* gives the average rank of each configuration for each one of the three inference schemes. *Pos* shows the absolute position in the ranking. Frd reports the p-value obtained with the multiple-comparison Friedman test performed for each inference methodology.

	wavg			**awin**			**nfit**		
	Cp	**C1**	**C2**	**Cp**	**C1**	**C2**	**Cp**	**C1**	**C2**
bal	1212	827	1580	114	128	122	75	105	60
bpa	1440	1714	1623	73	103	64	39	63	30
gls	2799	2551	3484	62	81	60	36	58	27
h-s	3415	2456	4002	117	139	122	62	93	62
irs	480	1217	634	18	26	19	7	10	6
pim	2841	1976	3407	192	251	183	62	141	42
tao	111	153	117	19	19	19	14	15	12
thy	1283	1838	1487	37	49	36	11	14	9
veh	3732	1776	4717	332	428	431	147	221	136
wbcd	3130	2305	4299	138	161	145	28	70	31
wdbc	5412	2724	5243	276	406	271	101	211	102
wne	3686	3695	4529	95	137	103	26	58	27
Rank	1.67	1.75	2.58	1.50	2.92	1.58	1.75	3.00	1.25
Pos	1	2	3	1	3	2	2	3	1
Frd	0.0458			0.0010			0.00006		

The rule set sizes evolved with the different configurations were not statistically equivalent according to the Friedman's test at a significance level of 0.05. To detect the significant differences among configurations, we applied the Nemenyi test with $\alpha = 0.1$ (that is, the critical difference is CD=0.83). The Nemenyi test identified the following three significant differences: (i) with weighted average inference, $C2$ resulted in significantly larger rule sets than Cp; (ii) with action winner inference, $C1$ created significantly larger rule sets than the other two configurations; (iii) with most numerous and fittest rules inference, $C1$ built significantly larger models than the other two configurations. The last two points can be easily explained as follows. As $C1$ used a low value of $P_{\#}$, final populations contained more specific classifiers than populations created with Cp and $C2$. These two inference schemes only kept the classifiers that maximally matched an input instance in the final population. Thus, if classifiers were more specific, a larger number of them were set to the final population. On the other hand, with weighted average inference, the largest populations were obtained with $C2$. We hypothesize that this is because $C2$ created slightly general and accurate classifiers that coexisted in the population and partially covered the same training instances. Although the learning process pressured to obtain a minimum set of these classifiers, the evolved populations of $C2$ were bigger than those obtained with the other two inference schemes since these classifiers with partially overlapped conditions were maintained in the final population as they covered some training instances with maximum degree.

D.3.2 Sensitivity to Fitness Pressure

In Fuzzy-UCS, fitness pressure is determined by the parameter ν. This parameter biases the selection pressure toward the fittest classifiers. A few analyses have been conducted on the correct setting of this parameter in LCSs. Kharbat et al. (2005) showed that ν had a strong effect if proportionate selection was used in XCS and recommended to use values around 10 for this parameter. Similarly, Brown et al. (2007) empirically showed that $\nu = 10$ was an optimal setting for UCS in a set of artificial problems. Thus, in our experiments, we used $\nu = 10$.

As proceeds, we analyze the impact of decreasing ν on the quality of the models. To achieve this, tables D.3 and D.4 show the accuracies and sizes of the evolved models for $\nu = 1$ (C3) and $\nu = 5$ (C4). The statistical analysis indicated that the accuracy of the models was not equivalent on average according to the multiple-comparison Friedman's test. Therefore, we applied the Nemenyi test (at $\alpha = 0.10$) to detect significant differences among learners (the critical distance is CD $= 0.83$). The test identified that, for all inference schemes, Fuzzy-UCS evolved more accurate models with Cp than with $C3$. Besides, the models created with Cp held the first position of the ranking, and the models built with $C4$ held the second position of the ranking for any inference level. Thus, these results evidenced that larger values of ν yielded more accurate models. We applied a pairwise comparison between Cp and $C4$ according to a Wilcoxon signed-ranks test, which detected, with $p = 0.02$, that the models evolved with Cp were significantly more accurate than those created with $C4$.

Conclusions on the model sizes depend on the used inference scheme. For weighted average inference, $C3$ created significantly larger models than the other configurations, and $C4$ evolved significantly larger models than Cp according to a Bonferroni-Dunn test at $\alpha = 0.1$. So, the higher the fitness pressure was, the smaller the final models were. This evidenced that setting high values of ν is crucial to remove over-general classifiers in favor of highly-fit classifiers.

Table D.3: Comparison of the test accuracy obtained with the three types of inference and the three configurations which vary the fitness pressure ν. *Rank* gives the average rank of each configuration for each one of the three inference schemes. *Pos* shows the absolute position in the ranking. *Frd* reports the p-value obtained with the multiple-comparison Friedman test performed for each inference methodology.

	wavg			**awin**			**nfit**		
	Cp	**C3**	**C4**	**Cp**	**C3**	**C4**	**Cp**	**C3**	**C4**
bal	88.65	83.31	87.87	84.40	80.11	83.90	83.40	63.33	82.49
bpa	59.82	57.94	59.08	59.42	58.42	58.50	58.93	56.45	60.23
gls	60.65	55.55	58.23	57.21	52.60	56.92	57.43	48.45	56.06
h-s	81.33	83.52	82.03	80.78	81.89	81.53	78.11	78.59	78.53
irs	95.67	93.13	95.07	95.47	94.80	95.07	93.73	82.60	92.27
pim	74.88	71.30	74.61	74.11	72.07	73.38	74.32	71.81	73.80
tao	81.71	79.18	80.90	83.02	82.57	83.36	87.53	79.05	86.68
thy	88.18	78.67	85.88	89.49	87.42	89.02	91.25	86.18	89.11
veh	67.68	61.12	67.23	65.35	61.22	64.77	65.34	57.59	65.79
wbcd	96.01	95.33	95.50	95.73	94.90	95.42	95.29	92.39	94.40
wdbc	95.20	94.68	94.96	94.61	93.99	94.11	94.51	92.88	94.00
wne	94.12	93.27	94.49	94.86	94.98	95.20	91.82	85.29	93.92
Rank	1.25	2.83	1.92	1.42	2.75	1.83	1.42	2.83	1.75
Pos	1	3	2	1	3	2	1	3	2
Frd	0.00050			0.00380			0.00140		

Table D.4: Comparison of the model sizes obtained with the three types of inference and the three configurations which vary the fitness pressure ν. *Rank* gives the average rank of each configuration for each one of the three inference schemes. *Pos* shows the absolute position in the ranking. *Frd* reports the p-value obtained with the multiple-comparison Friedman test performed for each inference methodology.

	wavg			**awin**			**nfit**		
	Cp	**C3**	**C4**	**Cp**	**C3**	**C4**	**Cp**	**C3**	**C4**
bal	1212	2371	1580	114	63	122	75	8	60
bpa	1440	2960	1623	73	47	64	39	12	30
gls	2799	4484	3484	62	52	60	36	17	27
h-s	3415	5469	4002	117	115	122	62	31	62
irs	480	1334	634	18	20	19	7	5	6
pim	2841	4166	3407	192	132	183	62	22	42
tao	111	149	117	19	16	19	14	5	12
thy	1283	2122	1487	37	32	36	11	7	9
veh	3732	5709	4717	332	242	431	147	84	136
wbcd	3130	4992	4299	138	137	145	28	26	31
wdbc	5412	5839	5243	276	255	271	101	92	102
wne	3686	5514	4529	95	97	103	26	29	27
Rank	1.08	3.00	1.92	2.33	1.25	2.42	2.67	1.17	2.17
Pos	1	3	2	2	1	3	3	1	2
Frd	0.00001			0.0052			0.0010		

For action winner inference, the statistical analysis only detected that the models evolved with $C3$ were significantly smaller than the models created with the other two configurations. This might be due to the presence of over-general classifiers with moderate fitness in the final populations, which had not been removed due to the poor genetic pressure toward highly fit classifiers. These classifiers replaced more specific but fitter ones in the final population. The same behavior could be observed for most numerous and fittest rules inference, where Fuzzy-UCS with configuration $C3$ created significantly smaller models than with the other two configurations.

D.3.3 Sensitivity to the GA

In this section, we analyze the sensitivity of Fuzzy-UCS to the parameters concerning the genetic algorithm application: θ_{GA}, θ_{del}, and θ_{sub}. The parameter θ_{GA} is a threshold that controls the application period of the GA on the different correct sets [C] of the system. That is, a correct set will receive a genetic event if the average time since the last application of the GA on this correct set exceeds θ_{GA}. If we want to maximize the genetic discovery, and so, the learning rate, θ_{GA} should be set to zero. In this case, a correct set would receive a genetic event every time it is activated. However, note that the parameters of new classifiers are incrementally updated as these classifiers participate in successive correct sets. So, if we apply a genetic algorithm to each correct set, selection would be biased since there would be poorly evaluated classifiers in the niches. Moreover, this would also imply the generation of a large number of new classifiers with poorly evaluated parameters. For this reason, θ_{GA} should be set to a higher value in real-world problems. In the configuration used in the paper, we set $\theta_{GA} = 50$, since this corresponds to the standard value used for the equivalent parameter in XCS and UCS (Bernadó-Mansilla et al., 2002; Bernadó-Mansilla and Garrell, 2003; Butz, 2006; Orriols-Puig and Bernadó-Mansilla, 2008b; Dixon et al., 2002, 2004; Fu et al., 2001; Wilson, 2000).

The values of the θ_{del} and θ_{sub} parameters are usually determined by θ_{GA}. If we consider that the classifiers in a niche need to receive an average of θ_{GA} parameter updates before going through a genetic event, intuitively this should also apply for deletion and subsumption. For this reason, in the configuration used in chapter 8, we set $\theta_{del} = \theta_{sub} = \theta_{GA}$.

To analyze the sensitivity of Fuzzy-UCS to this set of parameters, we performed the following experiments. In configurations $C5$ and $C6$, we incremented θ_{GA}, θ_{del}, and θ_{sub} to 100 and 200 respectively. As we decreased the application rate of the genetic algorithm, we expected to obtain a lower accuracy in the final models. Moreover, to confirm the stability of the system, we ran $C7$ and $C8$, two other configurations of Fuzzy-UCS. $C7$ equaled $C5$ except for the number of iterations, $numIter$=200 000. $C8$ equaled $C6$ except for the number of iterations, $numIter$=400 000. Therefore, we guaranteed that the number of genetic events received by the niches with configurations $C7$ and $C8$ was equivalent to the number of genetic events received with configuration Cp. Thus, we expected to obtain similar results.

Tables D.5 and D.6 show the accuracies and sizes of the models for the different configurations. The multiple-comparison Friedman's test rejected the null hypothesis that all the configurations performed the same on average for a particular inference scheme at 95% confidence level. As configuration Cp is the best ranked in all cases, we used the Bonferroni-Dunn test to detect which configurations performed worse than Cp at $\alpha = 0.10$ (the critical difference is $CD = 1.44$). The statistical test detected that: (i) with weighted average and most numerous and fittest rules, inferences Cp, C7, and C8 performed equivalently, whilst C5 and

Table D.5: Comparison of the test accuracy obtained with the three types of inference and the five configurations varying θ_{GA}, θ_{del}, and θ_{sub}. *Rank* gives the average rank of each configuration for each one of the three inference schemes. *Pos* shows the absolute position in the ranking. *Frd* reports the p-value obtained with the multiple-comparison Friedman test performed for each inference methodology.

	wavg					**awin**					**nfit**				
	Rp	**C5**	**C6**	**C7**	**C8**	**Rp**	**C5**	**C6**	**C7**	**C8**	**Rp**	**C5**	**C6**	**C7**	**C8**
bal	88.65	86.88	86.35	88.52	88.50	84.40	83.95	82.82	84.25	83.33	83.40	82.46	80.85	83.59	83.42
bpa	59.82	58.38	59.18	59.76	59.17	59.42	57.99	57.57	58.64	57.16	58.93	58.76	60.75	57.32	57.07
gls	60.65	56.78	55.75	61.79	64.17	57.21	56.57	55.59	57.30	56.63	57.43	54.77	53.50	57.91	58.96
h-s	81.33	81.04	81.81	80.75	79.79	80.78	80.89	80.78	80.40	80.71	78.11	78.52	79.30	77.57	77.08
irs	95.67	94.73	94.40	94.87	95.00	95.47	95.73	94.73	95.20	95.67	93.73	93.40	92.60	93.73	94.40
pim	74.88	74.00	72.17	74.75	74.74	74.11	73.17	71.81	73.83	74.00	74.32	73.71	72.78	73.25	72.62
tao	81.71	81.67	80.98	81.79	82.05	83.02	82.91	82.17	82.95	82.97	87.53	84.96	82.13	87.54	86.97
thy	88.18	86.89	85.41	88.18	89.65	89.49	89.45	87.93	89.81	90.00	91.25	89.34	88.28	90.20	91.74
veh	67.68	65.64	64.10	67.89	67.41	65.35	64.75	63.40	65.92	65.23	65.34	64.11	62.13	65.82	64.69
wbcd	96.01	95.65	95.14	95.82	95.74	95.73	95.43	94.96	95.76	95.92	95.29	94.73	94.26	95.29	95.14
wdbc	95.20	95.12	94.96	95.04	95.28	94.61	94.41	93.99	94.87	94.71	94.51	94.25	94.09	94.49	94.15
wne	94.12	93.50	94.73	95.12	94.91	94.86	94.08	94.46	95.24	94.93	91.82	90.33	90.88	92.09	92.86
Rnk	1.92	4.00	4.33	2.33	2.42	2.13	3.42	4.63	2.25	2.58	2.13	3.58	4.17	2.29	2.83
Pos	1	4	5	2	3	1	4	5	2	3	1	4	5	2	3
Frd	0.00014					0.00035					0.006				

C6 presented significantly poorer results; and (ii) with action winner, Cp, C5, C7, and C8 had the same accuracy on average, while C6 showed the poorest results. Further statistical analysis by means of pairwise comparisons supported these conclusions and, moreover, detected that C5 degraded the results obtained with Cp, C7, and C8 with action winner inference (see table D.7). Therefore, all this statistical study supported the initial hypothesis: as the number of genetic events decreases, the evolved models are less accurate.

The statistical analysis on the model sizes only identified significant differences for weighted average inference. In this case, the post-hoc Bonferroni-Dunn test detected that configuration $C6$ evolved larger models than the other configurations. This is because, with configuration $C6$, the correct sets received the lowest number of genetic events; therefore, the population had less diversity.

D.3.4 Sensitivity to Deletion

The deletion mechanism designed for Fuzzy-UCS was inspired by the initial deletion procedure of XCS (Kovacs, 1999). This schema increases the probability of deletion of experienced classifiers whose fitness is less than δ times the average fitness of the population. So, varying δ results in changing the pressure toward deletion of classifiers with low fitness. Nonetheless, recent studies have shown that XCS is not sensitive to the settings of δ (Kovacs and Bull, 2007). To confirm this statement, we ran XCS with $\delta = 1$ (configuration $C9$).

Table D.8 and D.9 compare the models accuracies and sizes of Fuzzy-UCS with configurations Cp and $C9$. We applied a pairwise comparison between the two configurations for each inference

Table D.6: Comparison of the model sizes obtained with the three types of inference and the five configurations varying θ_{GA}, θ_{del}, and θ_{sub}. *Rank* gives the average rank of each configuration for each one of the three inference schemes. *Pos* shows the absolute position in the ranking. *Frd* reports the p-value obtained with the multiple-comparison Friedman test performed for each inference methodology.

	wavg					**awin**					**nfit**				
	Rp	**C5**	**C6**	**C7**	**C8**	**Rp**	**C5**	**C6**	**C7**	**C8**	**Rp**	**C5**	**C6**	**C7**	**C8**
bal	1212	1233	1164	1096	1002	114	117	119	115	114	75	71	63	80	84
bpa	1440	1320	934	1519	1607	73	60	52	74	73	39	24	20	43	43
gls	2799	2684	2528	2835	2926	62	62	60	59	59	36	31	28	39	41
h-s	3415	3505	3273	3449	3396	117	133	137	113	107	62	68	67	59	58
irs	480	540	378	495	482	18	18	18	17	17	7	8	8	7	7
pim	2841	2539	2072	2707	2686	192	179	161	193	181	62	48	41	79	87
tao	111	143	117	107	101	19	18	17	19	19	14	13	10	14	14
thy	1283	1166	780	1266	1259	37	37	34	38	36	11	11	11	10	10
veh	3732	3859	3981	3581	3498	332	319	310	326	317	147	143	116	169	199
wbcd	3130	3184	2477	3097	3111	138	148	150	138	140	28	33	36	28	28
wdbc	5412	5343	5037	5415	5412	276	279	272	275	269	101	88	83	112	115
wne	3686	3568	3241	3746	3764	95	101	100	96	95	26	29	30	25	24
Rank	3.42	3.42	1.75	3.42	3.00	3.17	3.50	2.83	3.33	2.17	3.17	3.50	2.83	3.33	2.17
Pos	4	4	1	4	2	3	5	2	4	1	3	5	2	4	1
Frd	0.0368					0.1466					0.6289				

Table D.7: Pairwise comparison of the test accuracy of Fuzzy-UCS obtained with the three types of inference and the five configurations varying θ_{GA}, θ_{del}, and θ_{sub} by means of a Wilcoxon signed-ranks test.

	wavg					**awin**					**nfit**				
	Cp	**C5**	**C6**	**C7**	**C8**	**Cp**	**C5**	**C6**	**C7**	**C8**	**Cp**	**C5**	**C6**	**C7**	**C8**
Cp		.002	.008	.695	.938		.012	.003	.875	.480		.006	.034	.424	.530
C5	⊖		.182	.010	.012	⊖		.004	.041	.347	⊖		.100	.084	.182
C6	⊖	−		.012	.015	⊖	⊖		.003	.005	⊖	−		.060	.100
C7	−	⊕	⊕		.938	+	⊕	⊕		.327	−	+	+		1.00
C8	−	⊕	⊕	−		−	+	⊕	+		−	+	+	−	

scheme according to a Wilcoxon signed-ranks test (the approximate p-value is provided in the last row of the tables). The null hypothesis that the results obtained with both configurations were equal on average could not be rejected. This supported the empirical conclusions extracted by Kovacs and Bull (2007), which highlighted the robustness of XCS (and Fuzzy-UCS in our case) to the parameter δ.

Table D.8: Comparison of the test accuracy obtained with the three types of inference and the two configurations which vary the deletion pressure δ. *Rank* gives the average rank of each configuration for each one of the three inference schemes. *Pos* shows the absolute position in the ranking. *PW* reports the p-value obtained with the pairwise Wilcoxon signed-ranks test performed for each inference methodology.

	wavg		**awin**		**nfit**	
	Cp	**C9**	**Cp**	**C9**	**Cp**	**C9**
bal	88.65	88.68	84.40	84.30	83.40	82.91
bpa	59.82	59.61	59.42	58.37	58.93	58.32
gls	60.65	59.85	57.21	56.70	57.43	58.21
h-s	81.33	80.81	80.78	80.78	78.11	79.44
irs	95.67	95.13	95.47	95.60	93.73	94.40
pim	74.88	74.70	74.11	74.19	74.32	74.11
tao	81.71	81.64	83.02	83.11	87.53	87.79
thy	88.18	89.04	89.49	90.29	91.25	89.47
veh	67.68	66.50	65.35	65.84	65.34	65.59
wbcd	96.01	95.59	95.73	95.52	95.29	95.03
wdbc	95.20	95.09	94.61	94.43	94.51	94.19
wne	94.12	94.58	94.86	94.97	91.82	92.94
Rank	1.25	1.75	1.54	1.46	1.5	1.5
Pos	1	2	2	1	1.5	1.5
PW	0.1099		0.8311		0.7334	

Table D.9: Comparison of the model sizes obtained with the three types of inference and the three configurations which vary the deletion pressure δ. *Rank* gives the average rank of each configuration for each one of the three inference schemes. *Pos* shows the absolute position in the ranking. *PW* reports the p-value obtained with the pairwise Wilcoxon signed-ranks test performed for each inference methodology.

	wavg		**awin**		**nfit**	
	Cp	**C9**	**Cp**	**C9**	**Cp**	**C9**
bal	1212	1310	114	109	75	67
bpa	1440	1437	73	74	39	40
gls	2799	2869	62	60	36	35
h-s	3415	3450	117	116	62	60
irs	480	492	18	17	7	7
pim	2841	2765	192	188	62	66
tao	111	107	19	19	14	14
thy	1283	1276	37	37	11	11
veh	3732	3741	332	321	147	139
wbcd	3130	3385	138	135	28	28
wdbc	5412	5439	276	277	101	99
wne	3686	3808	95	97	26	25
Rank	1.33	1.67	1.75	1.25	1.75	1.25
Pos	1	2	2	1	2	1
PW	0.064		0.0977		0.1719	

D.4 Summary and Conclusions

The study performed in this chapter empirically showed that there are two key parameters to guarantee the success of Fuzzy-UCS: generalization in initialization ($P_{\#}$) and fitness pressure (ν). Specifically, the generality in the initial population has to be high enough to let the genetic algorithm take place, as suggested by Butz et al. (2001). Moreover, the fitness pressure should be high enough to ensure a strong and reliable pressure toward the fittest classifiers in the population. On the other hand, changing the setting of the other parameters appears to have little effect on the model's quality.

We acknowledge that better results could be individually obtained if we tuned Fuzzy-UCS for each particular problem. Nonetheless, we are interested in robust systems that perform well on average. For this reason we did not consider to tune the system for each problem in the experiments conducted in chapter 8.

Bibliography

C. Aggarwal, editor. *Data streams: Models and algorithms.* Springer, 2007.

J. S. Aguilar-Ruiz, J. C. Riquelme, and M. Toro. Evolutionary learning of hierarchical decision rules. *IEEE Transactions on Systems, Man, and Cybernetics—Part B*, 33(2):324–331, 2003.

J. S. Aguilar-Ruiz, R. Giráldez, and J. C. Riquelme. Natural encoding for evolutionary supervised learning. *IEEE Transactions on Evolutionary Computation*, 11(4):466–479, 2007.

D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1): 37–66, 1991.

R. Alcalá, J. Casillas, O. Cordón, and F. Herrera. Building fuzzy graphs: Features and taxonomy of learning for non-grid-oriented fuzzy rule-based systems. *Journal of Intelligent and Fuzzy Systems*, 11(3-4):99–119, 2001.

J. Alcalá-Fdez, F. Herrera, F. Márquez, and A. Peregrín. Increasing fuzzy rules cooperation based on evolutionary adaptive inference systems: Research articles. *International Journal in Intelligent Systems*, 22(9):1035–1064, 2007. ISSN 0884-8173. doi: http://dx.doi.org/10.1002/int.v22:9.

J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera. KEEL: A software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, doi=10.1007/s00500-008-0323-y, 2008.

D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The traveling salesman problem: A computational Study.* Princeton University Press, Princeton, NJ, USA, 2006.

A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nishizawa, J. Rosenstein, and J. Widom. STREAM: The stanford stream data manager demonstration description - short overview of system status and plans. In *SIGMOD'03: Proceedings of the ACM International Conference on Management of Data*, 2003.

A. Asuncion and D. J. Newman. *UCI Machine Learning Repository: [http://www.ics.uci.edu/∼mlearn/MLRepository.html].* University of California, 2007.

J. Bacardit. *GAssist Source Code: http://www.asap.cs.nott.ac.uk/∼jqb/PSP/GAssist-Java.tar.gz*, 2007.

J. Bacardit. *Pittsburgh genetic-based machine learning in the data mining era: Representations, generalization and run-time.* PhD thesis, Ramon Llull University, Barcelona, Catalonia, Spain, Barcelona, 2004.

J. Bacardit and M. V. Butz. Data mining in learning classifier systems: Comparing XCS with GAssist. In *Proceedings of the 7th International Workshop on Learning Classifier Systems.* Springer-Verlag, 2004.

T. Bäck. Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 57–62, 1994.

T. Bäck. Generalized convergence models for tournament- and (mu, lambda)-selection. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 2–8. Morgan Kaufmann Publishers Inc., 1995. ISBN 1-55860-370-0.

T. Bäck. *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, USA, 1996.

J. E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, pages 101–111, 1985.

A. Bardossy and L. Duckstein. *Fuzzy rule-based modeling with applications to geophysical, biological, and engineering systems.* CRC Press, Inc., Boca Raton, FL, USA, 1995. ISBN 0849378338.

G. Batista, R. C. Prati, and M. C. Monrad. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations Newsletter, special issue on learning from imbalanced datasets*, 6(1):20–29, 2004.

E. Bernadó-Mansilla and J. M. Garrell. Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

E. Bernadó-Mansilla and T. Ho. Domain of competence of XCS classifier system in complexity measurement space. *IEEE Transactions on Evolutionary Computation*, 9(1):1–23, 2005.

E. Bernadó-Mansilla, X. Llorà, and J. Garrell. XCS and GALE: A comparative study of two learning classifier systems on data mining. In *Advances in Learning Classifier Systems*, volume 2321 of *LNAI*, pages 115–132. Springer, 2002.

E. Bernadó-Mansilla, T. Ho, and A. Orriols-Puig. *Data Complexity and Evolutionary Learning: Classifier's Behavior and Domain of Competence*, pages 115–134. Springer, 2006.

H. G. Beyer. Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation*, 3(3):311–347, 1996.

C. M. Bishop. *Pattern recognition and machine learning.* Springer, 2007. ISBN 0-387-31073-8.

T. Blickle and L. Thiele. A mathematical analysis of tournament selection. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 9–16, 1995.

T. Blickle and L. Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, 4(4):361–394, 1996.

A. Bonarini. Evolutionary learning of fuzzy rules: Competition and cooperation. In W. Pedrycz, editor, *Fuzzy Modelling: Paradigms and Practice*, pages 265–284. Norwell, MA: Kluwer Academic Press, 1996.

A. Bonarini and V. Trianni. Learning fuzzy classifier systems for multi-agent coordination. *Information Sciences: an International Journal*, 136(1-4):215–239, 2001. ISSN 0020-0255.

P. Bonelli and A. Parodi. An efficient classifier system and its experimental comparison with two representative learning methods on three medical domains. In *4th International Conference on Genetic Algorithms*, pages 288–295, 1991.

L. Booker. *Intelligent behavior as an adaptation to the task environment.* PhD thesis, University of Michigan, 1982.

G. E. P. Box. Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics*, 6:81–101, 1957.

G. E. P. Box and N. P. Draper. *Evolutionary operation. A method for increasing industrial productivity.* New York: Wiley, 1969.

H. J. Bremermann. Optimization through evolution and recombination. In *Self-Organizing Systems*. Pergamon Press, Oxford, U.K., 1962.

R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

G. Brown, T. Kovacs, and J. Marshall. UCSpv: Principled voting in UCS rule populations. In *GECCO'07: Proceedings of the 2007 Conference on Genetic and Evolutionary Computation*, pages 1774–1781, New York, NY, USA, 2007. ACM.

B. G. Buchanan. A (very) brief history of artificial intelligence. *AI Magazine*, 26(4):53–60, 2005.

J. J. Buckley and Y. Hayashi. Fuzzy neural networks: A survey. *Fuzzy Sets and Systems*, 66: 1–13, 1994.

J. J. Buckley and Y. Hayashi. Neural networks for fuzzy systems. *Fuzzy Sets and Systems*, 71: 265–276, 1995.

L. Bull. *Applications of learning classifier systems.* Springer Verlag, 2004.

L. Bull. Two simple learning classifier systems. In *Foundations of Learning Classifier Systems*, volume 183/2005, pages 63–89. Springer Berlin, 2005.

L. Bull and J. Hurst. ZCS redux. *Evolutionary Computation*, 10(2):185–205, 2002.

L. Bull and T. O'Hara. Accuracy-based neuro and neuro-fuzzy classifier systems. In *GECCO'02: Proceedings of the 2002 Genetic and Evolutionary Computation Conference*, pages 905–911, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1-55860-878-8.

L. Bull, E. Bernadó-Mansilla, and J. Holmes, editors. *Learning classifier systems in data mining.* Studies in Computational Intelligence. Springer, 2008.

M. Butz, D. E. Goldberg, P. L. Lanzi, and K. Sastry. Problem solution sustenance in XCS: Markov chain analysis of niche support distributions and the impact on computational complexity. *Genetic Programming and Evolvable Machines*, 8(1):5–37, 2007.

M. V. Butz. *Rule-based evolutionary online learning systems: A principled approach to LCS analysis and design*, volume 109 of *Studies in Fuzziness and Soft Computing.* Springer, 2006.

M. V. Butz and M. Pelikan. Analyzing the evolutionary pressures in XCS. In *GECCO'01: Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pages 935–942. San Francisco, CA: Morgan Kaufmann, 2001.

M. V. Butz and S. W. Wilson. An algorithmic description of XCS. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems: Proceedings of the Third International Workshop*, volume 1996 of *Lecture Notes in Artificial Intelligence*, pages 253–272. Springer, 2001.

M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson. How XCS evolves accurate classifiers. In *GECCO'01: Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pages 927–934. San Francisco, CA: Morgan Kaufmann, 2001.

M. V. Butz, D. E. Goldberg, and T. Tharankunnel. Analysis and improvement of fitness exploration in XCS: Bounding models, tournament selection, and bilateral accuracy. *Evolutionary Computation*, 11(3):239–277, 2003.

M. V. Butz, D. E. Goldberg, and P. L. Lanzi. Bounding learning time in XCS. In *GECCO'2004: Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, LNCS, pages 739–750. Springer, 2004a.

M. V. Butz, T. Kovacs, P. L. Lanzi, and S. W. Wilson. Toward a theory of generalization and learning in XCS. *IEEE Transactions on Evolutionary Computation*, 8(1):28–46, 2004b.

M. V. Butz, D. E. Goldberg, and P. L. Lanzi. Gradient descent methods in learning classifier systems: Improving XCS performance in multistep problems. *IEEE Transactions on Evolutionary Computation*, 9(5):452–473, 2005a.

M. V. Butz, D. E. Goldberg, and P. L. Lanzi. Extracted global structure makes local building block processing effective in XCS. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005)*, pages 655–662. ACM Press, 2005b.

M. V. Butz, K. Sastry, and D. E. Goldberg. Strong, stable, and reliable fitness pressure in XCS due to tournament selection. *Genetic Programming and Evolvable Machines*, 6(1):53–77, 2005c.

M. V. Butz, P. L. Lanzi, and S. W. Wilson. Hyper-ellipsoidal conditions in XCS: Rotation, linear approximation, and solution structure. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1457—1464, New York, NY, USA, 2006. ACM. ISBN 1-59593-186-4. doi: http://doi.acm.org/10.1145/1143997.1144237.

M. V. Butz, P. L. Lanzi, and S. W. Wilson. Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Transactions on Evolutionary Computation*, 12(3):355–376, 2008. doi: 10.1109/TEVC.2007.903551.

E. Cantú-Paz. Selection intensity in genetic algorithms with generation gaps. In *GECCO'00: Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, pages 911–918, 1999a.

E. Cantú-Paz. Migration policies and takeover times in parallel genetic algorithms. In *GECCO'99: Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pages 775–775, 1999b. also IlliGAL Report No. 99008.

B. Carse, T. Fogarty, and A. Munro. Evolving fuzzy rule based controllers using genetic algorithms. *Fuzzy Sets and Systems*, 80(3):273–293, 1996. ISSN 0165-0114.

D. R. Carvalho and A. A. Freitas. A genetic-algorithm for discovering small-disjunct rules in data mining. *Applied Soft Computing*, 2(2):75–88, 2002.

J. Casillas, O. Cordón, M. J. del Jesus, and F. Herrera. Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. *IEEE Transactions on Fuzzy Systems*, 13(1):13–29, 2005. ISSN 1063-6706. doi: 10.1109/TFUZZ.2004.839670.

J. Casillas, B. Carse, and L. Bull. Fuzzy-XCS: A Michigan genetic fuzzy system. *IEEE Transactions on Fuzzy Systems*, 15(4):536–550, 2007.

P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Knowledge Discovery and Data Mining*, pages 164–168, 1998.

N. V. Chawla, K. Bowyer, L. O. Hall, and W. Kegelmeyer. SMOTE: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer. SMOTEBoost: Improving prediction of the minority class in boosting. In *VIIth European Conference on Principles and Practice of Knowledge Discovery in Databases(PKDD0́3)*, pages 107–119. Springer-Verlag, 2003.

N. V. Chawla, N. Japkowicz, and A. Kolcz, editors. *Special issue on learning from imbalanced datasets*, volume 6. 2004.

O. Cordón and F. Herrera. A three-stage evolutionary process for learning descriptive and approximate fuzzy-logic-controller knowledge bases from examples. *International Journal of Approximate Reasoning*, 17(4):369–407, 1997.

O. Cordón, M. del Jesús, and F. Herrera. A proposal on reasoning methods in fuzzy rule-based classification systems. *International Journal of Approximate Reasoning*, 20(1):21–45, 1999.

O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena. *Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases*, volume 19 of *Advances in Fuzzy Systems—Aplications and Theory*. World Scientific, 2001a.

O. Cordón, F. Herrera, and P. Villar. Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. *IEEE Transactions on Fuzzy Systems*, 9(4): 667–674, August 2001b.

K. Crockett, Z. Bandar, and D. Mclean. On the optimization of t-norm parameters within fuzzy decision trees. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007)*, pages 103–108, 2007. doi: 10.1109/FUZZY.2007.4295348.

K. A. Crockett, Z. Bandar, J. Fowdar, and J. O'Shea. Genetic tuning of fuzzy inference within fuzzy classifier systems. *Expert Systems*, 23:63–82, 2006. doi: doi:10.1111/j.1468-0394.2006.00325.x.

H. H. Dam, H. A. Abbass, and C. Lokan. Be real! XCS with continuous-valued inputs. In *GECCO'05: In Proceedings of the 2005 Genetic and Evolutionary Computation Conference workshop program*, pages 85–87, Washington, D.C., USA, 2005. ACM Press.

C. Darwin. *The origin of species.* see online version at www.literature.org, 1859.

K. Deb. Genetic algorithms in multimodal function optimization. In *Master Thesis and TCGA Report No. 89002. Tuscaloosa, AL: Department of Engineering Mechanics, University of Alabama*, 1989.

M. J. del Jesus, F. Hoffmann, L. J. Navascués, and L. Sánchez. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. *IEEE Transactions on Fuzzy Systems*, 12 (3):296–308, 2004.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

A. V. den Bosch, T. Weijters, and J. V. den Herik. When small disjuncts abound, try lazy learning: A case study. In *Proceedings Seventh BENELEARN Conference*, pages 109–118, 1997.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

P. W. Dixon, D. W. Corne, and M. J. Oates. A preliminary investigation of modified XCS as a generic data mining tool. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems, 4th International Workshop*, volume 2321 of *Lecture Notes in Computer Science*, pages 133–150. Springer, 2002.

P. W. Dixon, D. W. Corne, and M. J. Oates. *A ruleset reduction algorithm for the XCSI learning classifier system*, volume 2661/2003 of *Lecture Notes in Computer Science*, pages 20–29. Springer, 2004. ISBN 978-3-540-20544-9.

J. Drugowitsch. *Design and analysis of learning classifier systems: A probabilistic approach.* Springer, 2008.

J. Drugowitsch and A. M. Barry. A formal framework and extensions for function approximation in learning classifier systems. *Machine Learning*, 70(1):45–88, 2008.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification.* Wiley-Interscience, 2nd edition, 2000.

O. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

T. Fawcett. PRIE: A system for generating rulelists to maximize ROC performance. *Data Mining and Knowledge Discovery*, 17(2):207–224, 2008. doi: 10.1007/s10618-008-0089-y.

E. A. Feigenbaum and J. Feldman, editors. *Computers and Thought.* AAAI press, 1995.

R. Fisher. *Statistical methods and scientific inference.* Hafner Publishing Co, New York, 2nd edition, 1959.

D. B. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.

D. B. Fogel. *On the organization of intellect.* PhD thesis, 1964.

D. B. Fogel, A. J. Owens, and M. J. Walsh. *Artificial intelligence through simulated evolution.* Wiley, New York, USA, 1966.

E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the 15th International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, San Francisco, CA, 1998.

W. Frawley, G. Piatetsky-Shapiro, and C. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, pages 213–228, 1992.

A. Freitas. *Data mining and knowledge discovery with evolutionary algorithms.* Springer-Verlag, 2002.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

R. M. Friedberg. A learning machine: part I. *IBM Journal*, 2:2–13, 1958.

R. M. Friedberg, B. Dunham, and J. H. North. A learning machine: part II. *IBM Journal*, 3: 282–287, 1959.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11:86–92, 1940.

C. Fu, S. W. Wilson, and L. Davis. Studies of the XCSI classifier system on a data mining problem. In *GECCO'01: Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pages 985–993, San Francisco, CA, USA, 2001. Morgan Kaufmann.

T. Furuhashi, K. Nakaoka, and Y. Uchikawa. Suppression of excess fuzziness using multiple fuzzy classifier systems. In *Proceedings of the 3th IEEE International Conference on Fuzzy Systems*, pages 411–414. Morgan Kaufmann, 1994.

J. Gama and M. M. Gaber, editors. *Learning from data streams.* Springer, 2007.

S. García and F. Herrera. Evolutionary under-sampling for classification with imbalanced data sets: Proposals and taxonomy. *Evolutionary Computation*, 2008.

A. Giordana and F. Neri. Search-intensive concept induction. *Evolutionary Computation*, 3(4): 375–419, 1995.

H. Goksu, P. Pigg, and V. Dixit. Music Composition Using Genetic Algorithms (GA) and Multilayer Perceptrons (MLP). In *Advances in Natural Computation*, volume 3612 of *Lectures Notes in Computer Science*, pages 1242–1250, 2005.

D. E. Goldberg. *The design of innovation: Lessons from and for competent genetic algorithms.* Kluwer Academic Publishers, 1 edition, 2002.

D. E. Goldberg. Controlling dynamic systems with genetic algorithms and rule learning. In *Proceedings of the 4th Yale Workshop on Applications and Adaptive Systems Theory*, pages 91–97, 1985a.

D. E. Goldberg. Dynamic system control using rule learning and genetic algorithms. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, volume 1, pages 588–592, 1985b.

D. E. Goldberg. Computer-aided gas pipeline operation using genetic algorithms and rule learning. part I: Genetic algorithms in pipeline optimization. *Engineering with Computers*, pages 35–45, 1987a.

D. E. Goldberg. Computer-aided gas pipeline operation using genetic algorithms and rule learning. part II: Rule learning control of a pipeline under normal and abnormal conditions. *Engineering with Computers*, pages 47–58, 1987b.

D. E. Goldberg. *Genetic algorithms in search, optimization & machine learning.* Addison Wesley, 1 edition, 1989a.

D. E. Goldberg. Sizing populations for serial and parallel genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 70–79, 1989b.

D. E. Goldberg. *Computer-aided gas pipeline operation using genetic algorithms and rule learning.* PhD thesis, University of Michigan, Ann Arbor, MI, 1983.

D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, pages 69–93, 2003.

D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

D. E. Goldberg and M. Rudnick. Genetic algorithms and the variance of fitness. *Complex Systems*, 5(3):265–278, 1991.

D. E. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1989.

D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.

D. E. Goldberg, D. Thierens, and K. Deb. Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers*, 32(1):10–16, 1993.

D. E. Goldberg, K. Sastry, and T. Latoza. On the supply of building blocks. In *GECCO'01: Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pages 336–342. Springer, 2001.

D. E. Goldberg, K. Sastry, and X. Llorà. Toward routine billion-variable optimization using genetic algorithms: Short communication. *Complexity*, 12(3):27–29, 2007. ISSN 1076-2787. doi: http://dx.doi.org/10.1002/cplx.v12:3.

A. González and R. Pérez. SLAVE: A genetic learning system based on an iterative approach. *IEEE Transactions on Fuzzy Systems*, 7(2):176–191, 1999.

D. P. Greene. Automated knowledge acquisition: Overcoming the expert systems bottleneck. In *Proceedings of the Seventh International Conference on Information Systems*, pages 107–117, Pittsburgh, PA, 1987. Lawrence Erlbaum Assoc.

D. P. Greene. *Inductive knowledge acquisition using genetic adaptive search.* PhD thesis, Pittsburgh, PA, 1992.

D. P. Greene and S. E. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13:229–257, 1993.

D. P. Greene and S. F. Smith. A genetic system for learning models of consumer choice. In *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*, pages 217–223, Boston, MA, 1987. Morgan Kaufmann.

J. J. Grefenstette and J. E. Baker. How genetic algorithms work: A critical look at implicit parallelism. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 20–27, 1989.

J. J. Grefenstette and J. Fitzpatrick. Genetic search with approximate function evaluations. In *International Conference on Genetic Algorithms and their Applications*, pages 112–120, 1992.

J. W. Grzymala-Busse, L. K. Goodwin, and W. J. Grzymala-Busse. An approach to imbalanced data sets based on changing rule strength. In *Learning from Imbalanced Data Sets: Papers from the AAAI Workshop*, pages 69–74, 2000.

H. Han, W. Y. Wang, and B. H. Mao. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *ICIC'05: Proceedings of the 2005 International Conference on Intelligent Computing*, pages 878–887. Springer-Verlag, 2005.

G. Harik. Linkage learning via probabilistic modeling in the ECGA. Technical report, Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign (IlliGAL Report No. 99010), 1999.

G. Harik. *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms.* PhD thesis, University of Michigan, Ann Arbor, 1997. Also available as IlliGAL Report 97005.

G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999. ISSN 1063-6560. doi: http://dx.doi.org/10.1162/evco.1999.7.3.231.

F. Herrera. Genetic fuzzy systems: Taxonomy and current research trends and prospects. *Evolutionary Intelligence*, 1:27–46, 2008. doi: 10.1007/s12065-007-0001-5.

S. Hettich and S. D. Bay. *The UCI KDD Archive [http://kdd.ics.uci.edu].* Irvine, CA: University of California, Department of Information and Computer Science, 1999.

T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002. ISSN 0162-8828.

J. H. Holland. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, 3:297–314, 1962.

J. H. Holland. Nonlinear environments permitting efficient adaptation. In *Computer and Information Sciences II.* New York: Academic, 1967.

J. H. Holland. Processing and processors for schemata. In E. L. Jacks, editor, *Associative information processing*, pages 127–146. New York: American Elsevier, 1971.

J. H. Holland. *Adaptation in natural and artificial systems.* The University of Michigan Press, 1975.

J. H. Holland. Adaptation. In R. Rosen and F. Snell, editors, *Progress in Theoretical Biology*, volume 4, pages 263–293. New York: Academic Press, 1976.

J. H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA., 2nd edition, 1992.

J. H. Holland and J. S. Reitman. Cognitive systems based on adaptive algorithms. In D. Waterman and F. Hayes-Roth, editors, *Pattern-directed inference systems*, pages 313–329. Academic Press, San Diego, USA, 1978.

J. Holmes. Differential negative reinforcement improves classifier system learning rate in two-class problems with unequal base rates. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 635–642. Morgan Kaufmann, 1998.

R. C. Holte, L. E. Acker, and B. W. Porter. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 813–818, 1989.

A. Homaifar and E. McCormick. Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(2): 129–139, 1995. ISSN 1063-6706. doi: 10.1109/91.388168.

J. Horn. *The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations.* PhD thesis, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois at Urbana Champaign, Urbana Champaign 117, 1997.

H. Ishibuchi and T. Yamamoto. Rule weight specification in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 13(4):428–435, 2005.

H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka. Selection fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, 3(3):260–270, 1995.

H. Ishibuchi, T. Murata, and I. B. Türkşen. Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. *Fuzzy Sets and Systems*, 89(2):135–150, 1997. ISSN 0165-0114. doi: http://dx.doi.org/10.1016/S0165-0114(96)00098-X.

H. Ishibuchi, T. Nakashima, and T. Morisawa. Voting in fuzzy rule-based systems for pattern classification problems. *Fuzzy Sets and Systems*, 103(2):223–238, 1999a.

H. Ishibuchi, T. Nakashima, and T. Murata. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 29(5):601–618, 1999b.

H. Ishibuchi, T. Yamamoto, and T. Murata. Hybridization of fuzzy GBML approaches for pattern classification problems. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 35(2):359–365, 2005.

B. Jahne, H. Haussecker, and P. Geissler, editors. *Handbook of Computer Vision and Applications*, volume 1-3. Academic Press, 1999.

C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13(2-3):189–228, 1993. ISSN 0885-6125.

N. Japkowicz and S. Stephen. The class imbalance problem: Significance and strategies. In *International Conference on Artificial Intelligence (IC-AI'00)*, volume 1, pages 111–117, 2000.

N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analisis*, 6(5):429–450, November 2002.

N. Japkowicz and J. Taeho. Class imbalances versus small disjuncts. *SIGKDD Explorations Newsletter, special issue on learning from imbalanced datasets*, 6(1):40–49, June 2004.

J. H. Jo and T. H. Ahn. Reviewing the use of genetic algorithms in a real-time computer game. In *Artificial Intelligence and Applications*, 2002.

G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, 1995.

K. A. D. Jong. *Evolutionary computation: A unified approach.* MIT Press, Cambridge MA, 2006.

K. A. D. Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

K. A. D. Jong and W. M. Spears. Learning concept classification rules using genetic algorithms. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 651–656. Sidney, Australia, 1991.

K. A. D. Jong, W. M. Spears, and D. Gordon. Using genetic algorithms for concept learning. *Genetic Algorithms for Machine Learning, A Special Issue of Machine Learning*, 13(2–3): 161–188, 1993.

C. M. Karat, J. Vergo, and D. Nahamoo. Conversational interface technologies. In *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, pages 169–186. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 2003. ISBN 0-8058-3838-4.

C. Karr. Genetic algorithms for fuzzy controllers. *AI Expert*, 6(2):26–33, 1991. ISSN 0888-3785.

F. Kharbat, L. Bull, and M. Odeh. Revisiting genetic selection in the XCS learning classifier system. In *Congress on Evolutionary Computation*, pages 2061–2068, Edinburgh, UK, 2-5 September 2005. IEEE.

D. Kim, Y. S. Choi, and S. Y. Lee. An accurate cog defuzzifier design using lamarckian co-adaptation of learning and evolution. *Fuzzy Sets and Systems*, 130:207–225, 2002. doi: doi: 10.1016/S0165-0114(01)00167-1.

H. Kitano. Empirical studies of the speed of convergence of neural networks training using genetic algorithms. In *In Proceedings of the Eight National Conference in Artificial Intelligence*, pages 789–796, 1990.

G. J. Klir and B. Yuan. *Fuzzy sets and fuzzy logic–Theory and applications.* Prentice-Hall, 1995.

J. Korst and E. Aarts. *Simulated annealing and boltzmann machines.* Wiley-Interscience, New York, 1997.

T. Kovacs. Towards a theory of strong overgeneral classifiers. In W. Martin and W. M. Spears, editors, *Foundations of Genetic Algorithms*, volume 6, pages 165–184. Morgan Kaufmann, 2001.

T. Kovacs. XCS classifier system reliably evolves accurate, complete, and minimal representations for boolean functions. In Roy, Chawdhry, and Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 59–68, London, UK, 1997. Springer-Verlag.

T. Kovacs. Deletion schemes for classifier systems. In *GECCO'99: Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pages 329–336. Morgan Kaufmann, 1999.

T. Kovacs and L. Bull. Toward a better understanding of rule initialisation and deletion. In *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 2777–2780, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-698-1.

T. Kovacs and M. Kerber. What makes a problem hard for XCS? In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems: Third International Workshop*, pages 80–99. Springer-Verlag, 2001.

T. Kovacs and M. Kerber. High classification accuracy does not imply effective genetic search. In *GECCO'04: 2004 Genetic and Evolutionary Computation Conference*, pages 785–796, Seattle, WA, USA. 26-30 June, 2004. Springer, LNCS 3103.

J. R. Koza. Hierarchical genetic algorithms operation on populations of computer programs. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, volume 1, pages 768–774, 1989.

J. R. Koza. *Genetic programming: On the programming of computers by means of natural selection.* MIT Press, Cambridge, Massachusetts, 1992.

J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic programming IV: Routine human-competitive machine intelligence.* Kluwer Academic Publishers, 2003.

M. Kubat, R. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3):195–215, 1998.

P. L. Lanzi. Learning classifier systems from a reinforcement learning perspective. *Soft Computing — A Fusion of Foundations, Methodologies and Applications*, 6(3):162–170, 2002.

P. L. Lanzi. Extending the representation of classifier conditions part I: From binary to messy coding. In *GECCO'99: Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pages 337–344, Orlando (FL), 1999a. Morgan Kaufmann.

P. L. Lanzi. *Reinforcement Learning with Classifier Systems.* PhD thesis, Politecnico di Milano, 1999b.

P. L. Lanzi and A. Perrucci. Extending the representation of classifier conditions part II: From messy coding to S-expressions. In *GECCO'99: Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, volume 1, pages 345–352. Morgan Kaufmann, 1999.

P. L. Lanzi and S. W. Wilson. Using convex hulls to represent classifier conditions. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1481–1488, New York, NY, USA, 2006. ACM. ISBN 1-59593-186-4. doi: http://doi.acm.org/10.1145/1143997.1144240.

P. L. Lanzi, D. Loiacono, S. W. Wilson, and D. E. Goldberg. XCS with computed prediction in continuous multistep environments. In *The 2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2032–2039, 2005. doi: 10.1109/CEC.2005.1554945.

P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation.* Springer, 2002.

C. F. Lima, M. Pelikan, K. Sastry, M. V. Butz, D. E. Goldberg, and F. Lobo. Substructural neighborhoods for local search in the bayesian optimization algorithm. In *PPSN IX: Parallel Problem Solving from Nature*, pages 232–241, 2006.

C. Ling and C. Li. Data mining for direct marketing: Problems and solutions. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 1998.

Z. Liu, A. Liu, C. Wang, and Z. Niu. Evolving neural network using real coded genetic algorithm (GA) for multispectral image classification. *Future Generation Computer Systems*, 20(7): 1119–1129, 2004. ISSN 0167-739X.

X. Llorà and J. M. Garrell. Evolution of decision trees. In *CCIA'01: Fourth Catalan Conference on Artificial Intelligence*, pages 115–122. ACIA Press, 2001.

X. Llorà and S. W. Wilson. Mixed decision trees: Minimizing knowledge representation bias in LCS. In *GECCO'04: Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, pages 797–809. Springer-Verlag, LNCS 3103, 2004.

X. Llorà, R. Reddy, B. Matesic, and R. Bhargava. Towards better than human capability in diagnosing prostate cancer using infrared spectroscopic imaging. In *GECCO'07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 2098–2105, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4. doi: http://doi.acm.org/10.1145/1276958.1277366.

N. Macià, E. Bernadó-Mansilla, and A. Orriols-Puig. Preliminary approach on synthetic datasets generation for classification. In *In 2008 International Conference on Pattern Recognition*, 2008a. in press.

N. Macià, E. Bernadó-Mansilla, and A. Orriols-Puig. On the dimensions of data complexity through synthetic data sets. In *In Recent Advances in Artificial Intelligence Research and Development*. IOS Press, 2008b. in press.

N. Macià, A. Orriols-Puig, and E. Bernadó-Mansilla. Genetic-based synthetic data sets for the analysis of classifiers' behavior. In *HIS'08: Proceedings of the 2008 Hybrid Intelligent Systems Conference*, 2008c. in press.

S. M. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois at Urbana Champaign, Urbana Champaign 117, 1995.

E. H. Mamdani. Applications of fuzzy algorithm for control a simple dynamic plant. In *Proceedings IEEE*, pages 1585–1588, 1974.

F. A. Marquez, A. Peregrin, and F. Herrera. Cooperative evolutionary learning of linguistic fuzzy rules and parametric aggregation connectors for mamdani fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 15(6):1162–1178, 2007. ISSN 1063-6706. doi: 10.1109/TFUZZ.2007.904121.

J. McCharthy. What is artificial intelligence. Technical report, Stanford University Report, Stanford, CA, USA, 2007.

M. McInerney and A. P. Dhawan. Use of genetic algorithms with backpropagation in training of feedforward neural networks. In *In Proceedings of the IEEE International Conference on Neural Networks (ICCN'93)*, pages 203–208, San Francisco, USA, 1993.

D. Mellor. A first order logic classifier system. In *GECCO'05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 1819–1826, New York, NY, USA, 2005. ACM. ISBN 1-59593-010-8. doi: http://doi.acm.org/10.1145/1068009.1068318.

R. S. Michalski. On the quasi-minimal solution of the covering problem. In *FCIP'69: Proceedings of the 5th International Symposium on Information Processing*, volume A3, pages 125–128, Bled, Yugoslavia, 1969.

R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The AQ15 inductive learning system: An overview and experiments. Technical report, Intelligent Systems Group, ISG 86-20, UIUCDCS-R-86-1260, Department of Computer Science, University of Illinois, Urbana, 1986.

D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine learning, neural and statistical classification.* Ellis Horwood, 1994.

I. Mierswa. Controlling overfitting with multi-objective support vector machines. In *GECCO'07: Proceedings of the 2007 Conference on Genetic and Evolutionary Computation*, pages 1830–1837, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4. doi: http://doi.acm.org/10.1145/1276958.1277323.

B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9:193–212, 1995.

B. L. Miller and D. E. Goldberg. Optimal sampling for genetic algorithms. *Intelligent Engineering Systems through Artificial Neural Networks*, 6:291–297, 1996.

T. M. Mitchell. The discipline of machine learning. Technical report, Machine Learning Department, School of Computer Science, Carnegie Melon University, Pittsburgh, PA, USA, 2006.

T. M. Mitchell. *Machine learning.* McGraw Hill, 1997.

S. Morales-Ortigosa, A. Orriols-Puig, and E. Bernadó-Mansilla. Can evolution strategies improve learning guidance in XCS? Design and comparison with genetic algorithms based XCS. In *Advances in Artificial Intelligence*, volume in press. IOS press, 2008a.

S. Morales-Ortigosa, A. Orriols-Puig, and E. Bernadó-Mansilla. New crossover operator for evolutionary rule discovery in XCS. In *Proceedings of the 2008 Hybrid and Intelligent Systems Conference*, volume in press. IEEE, 2008b.

H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm I. Continuous parameter optimization. *Evolutionary Computation*, 1(1):25–49, 1993.

K. Nakaoka, T. Furuhashi, and Y. Uchikawa. A study on apportionment of credits of fuzzy classifier system for knowledge adquisition in large scale systems. In *Proceedings of the 3th IEEE International Conference on Fuzzy Systems*, pages 1797–1800. Morgan Kaufmann, 1994.

P. B. Nemenyi. *Distribution-free multiple comparisons.* PhD thesis, Princeton University, New Jersey, USA, 1963.

N. J. Nilson. *Introduction to Machine Learning. Draft of Incomplete Notes.* Electronic version, draft edition, 2005.

Y. Nomura, K. Ikebukuro, K. Yokoyama, T. Takeuchi, Y. Arikawa, S. Ohno, I. Karube, and M. Valenzuela-Rendón. Reinforcement learning in the fuzzy classifier system. *Expert Systems with Applications*, 14:237–247, 1998.

A. Nurnberger, C. Borgelt, and A. Klose. Improving naive bayes classifiers using neuro-fuzzy learning. In *Proceedings of the 1999 Conference on Neural Information Processing*, volume 1, pages 154–159, Perth, WA, Australia, 1999.

A. Orriols-Puig and E. Bernadó-Mansilla. Analysis of reduction algorithms for XCS classifier system. In *Recent Advances in Artificial Intelligence Research and Development*, number 113 in 1, pages 383–390. IOS Press, October 2004.

A. Orriols-Puig and E. Bernadó-Mansilla. The class imbalance problem in learning classifier systems: A preliminary study. In *GECCO'05: Proceedings of the 2005 Genetic and Evolutionary Computation Conference Workshop Program*, pages 74–78, Washington, D.C., USA, 25-29 June 2005a. ACM Press.

A. Orriols-Puig and E. Bernadó-Mansilla. The class imbalance problem in UCS classifier system: Fitness adaptation. In *Congress on Evolutionary Computation*, volume 1, pages 604–611, Edinburgh, UK, 2-5 September 2005b. IEEE.

A. Orriols-Puig and E. Bernadó-Mansilla. Bounding XCS parameters for unbalanced datasets. In *GECCO'06: Proceedings of the 2006 Genetic and Evolutionary Computation Conference*, pages 1561–1568. ACM Press, 2006a.

A. Orriols-Puig and E. Bernadó-Mansilla. The class imbalance problem in UCS classifier system: A preliminary study. In *Advances at the frontier of LCS*, pages 164–183. Springer, 2007.

A. Orriols-Puig and E. Bernadó-Mansilla. A further look at UCS classifier system. In *GECCO'06: Proceedings of the 2006 Genetic and Evolutionary Computation Conference Workshop Program*, page to appear, Seattle, W.A., USA, 08–12 July 2006b. ACM Press.

A. Orriols-Puig and E. Bernadó-Mansilla. Revisiting UCS: Description, fitness sharing and comparison with XCS. In *Advances at the Frontier of LCSs*. Springer, 2008a.

A. Orriols-Puig and E. Bernadó-Mansilla. Facetwise analysis of XCS for problems with class imbalances. *IEEE Transactions on Evolutionary Computation, submitted*, 2008b.

A. Orriols-Puig and E. Bernadó-Mansilla. Mining imbalanced data with learning classifier systems. In *Learning Classifier Systems in Data Mining*, pages 123–145. Springer, 2008a.

A. Orriols-Puig and E. Bernadó-Mansilla. Evolutionary rule-based systems for imbalanced datasets. *Soft Computing Journal*, doi=10.1007/s00500-008-0319-7, 2008b.

A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Fuzzy-UCS: Preliminary results. In *GECCO'07: Proceedings of the 2007 Genetic and Evolutionary Computation Conference Workshop Program*, volume 3, pages 2871–2874. ACM Press, 2007a.

A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Aprendizaje supervisado de reglas difusas mediante un sistema clasificador evolutivo estilo Michigan. In *Proceedings of the II Congreso Espa nol de Informática (CEDI 2007). I Jornadas sobre Algoritmos Evolutivos y Metaheurísticas(JAEM07)*, pages 171–178, 2007b.

A. Orriols-Puig, D. E. Goldberg, K. Sastry, and E. Bernadó-Mansilla. Modeling XCS in class imbalances: Population size and parameters' settings. In *GECCO'07: Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, volume 2, pages 1838–1845. ACM Press, 2007c.

A. Orriols-Puig, K. Sastry, P. Lanzi, D. Goldberg, and E. Bernadó-Mansilla. Modeling selection pressure in XCS for proportionate and tournament selection. In *GECCO'07: Proceedings of the 2007 Genetic and Evolutionary Computation Conference*, volume 2, pages 1846–1853. ACM Press, 2007d.

A. Orriols-Puig, E. Bernadó-Mansilla, N. Macià, and T. K. Ho. CMAPI: A complexity metrics API. *Journal of Machine Learning Research (submitted)*, 2008a.

A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Evolving fuzzy rules with UCS. In *Advances at the Frontier of LCSs*. Springer, 2008b.

A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Genetic-based machine learning systems are competitive for pattern recognition. *Evolutionary Intelligence*, 2008c. doi: 10.1007/s12065-008-0013-9.

A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. A comparative study of several classifiers in supervised learning. In *Learning Classifier Systems in Data Mining*, pages 205–230. Springer, 2008d.

A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Fuzzy-UCS: a Michigan-style learning fuzzy-classifier system for supervised learning. *IEEE Transactions on Evolutionary Computation*, 2008e. doi: 10.1109/TEVC.2008.925144.

A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. First approach toward on-line evolution of association rules with learning classifier systems. In *GECCO'08: Proceedings of the 2008 Genetic and Evolutionary Computation Conference Workshop Program*, pages 2031–2038, New York, NY, USA, 2008f. ACM. ISBN 978-1-60558-131-6. doi: http://doi.acm.org/10.1145/1388969.1389017.

A. Orriols-Puig, J. Casillas, and E. Bernadó-Mansilla. Approximate versus linguistic representation in fuzzy-ucs. In *HAIS'08: Proceedings of the 2008 International Workshop on Hybrid Artificial Intelligence Systems*, LNAI, (in press), 2008g.

A. Orriols-Puig, J. Casillas, and F. Martínez-López. Modelado causal en marketing mediante aprendizaje no supervisado de reglas de asociación difusas. In *ESTYLF'08: Proceedings of the XIV Congreso Español sobre Tecnologías y Lógica Fuzzy*, 2008h. Spanish version only.

J. Otero and L. Sánchez. Induction of descriptive fuzzy classifiers with the logitboost algorithm. *Soft Computing*, 10(9):825–835, 2006. ISSN 1432-7643.

G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 5:71–99, 1990.

A. Parodi and P. Bonelli. A new approach to fuzzy classifier systems. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 223–230. Morgan Kaufmann, 1993.

M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk. Reducing the misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 217–225, 1994.

W. Pedrycz. Fuzzy sets technology in knowledge discovery. *Fuzzy Sets and Systems*, 98(3): 279290, 1998.

M. Pelikan. *Hierarchical bayesian optimization algorithm: Toward a new generation of evolutionary algorithms*, volume 170 of *Studies in Computational Intelligence.* Springer, 2005.

M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The bayesian optimization algorithm. In *GECCO'09: Genetic and Evolutionary Computation Conference*, pages 525–532, 1999.

M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Linkage learning, estimation distribution, and bayesian networks. *Evolutionary Computation*, 8(3):314–341, 2000a.

M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. In *Proceedings of the American Control Conference*, pages 3289–3293, 2000b.

M. Pelikan, K. Sastry, and E. Cantú-Paz. *Scalable optimization via probabilistic modeling*, volume 33 of *Studies in Computational Intelligence.* Springer, 2006. ISBN 978-3-540-34953-2.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning.* MIT Press, 1998.

J. R. Quinlan. Discovering rules by induction from large collections of examples. In D. Michie, editor, *Expert Systems in the Micro Electronic Age*, Edinburgh, UK, 1979. Edinburgh University Press.

J. R. Quinlan. *C4.5: Programs for machine learning.* Morgan Kaufmann Publishers, San Mateo, California, 1995.

I. Rechenberg. *Cybernetic solution path of an experimental problem*, volume 1122. 1965.

I. Rechenberg. *Evolution strategie: Optimierung technischer systeme nach prinzipien der biologischen evolution.* Frommann-Holzboog, 1973.

R. L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

M. Robnik-Sikonja, D. Cukjati, and I. Kononenko. Comprehensible evaluation of prognostic factors and prediction of wound healing. *Artificial Intelligence in Medicine*, 29(1-2):25–38, 2003.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, pages 318–364, Cambridge, MA, 1986. MIT Press.

S. Russell and P. Norvig. *Artificial intelligence: A modern approach.* Series on Artificial Intelligence. Prentice Hall, 2nd edition, 2002.

L. Sánchez and I. Couso. Fuzzy random variables-based modeling with GA-P algorithms. In R. Yager and B. Bouchon-Menier, editors, *Information, Uncertainty and Fusion*, pages 245–256. Kluwer, 2000.

L. Sánchez and I. Couso. Advocating the use of imprecisely observed data in genetic fuzzy systems. *IEEE Transactions on Fuzzy Systems*, 15(4):551–562, 2007.

L. Sánchez and I. Couso. Learning with imprecise examples with GA-P algorithms. *Soft Computing*, 5(1–4):305–319, 1998.

L. Sánchez and J. Otero. Boosting fuzzy rules in classification problems under single-winner inference. *International Journal of Intelligent Systems*, 22(9):1021–1034, 2007. ISSN 0884-8173. doi: http://dx.doi.org/10.1002/int.v22:9.

L. Sánchez, I. Couso, and J. A. Corrales. Combining GP operators with SA search to evolve fuzzy rule based classifiers. *Information Sciences*, 136(1–4):175–191, 2001. ISSN 0020-0255.

L. Sánchez, I. Couso, and J. Casillas. Modeling vague data with genetic fuzzy systems under a combination of crisp and imprecise criteria. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making*, pages 346–353, 2007.

K. Sastry and D. E. Goldberg. Modeling tournament selection with replacement using apparent added noise. *Intelligent Engineering Systems Through Artificial Neural Networks*, 11:129–134, 2001.

K. Sastry and D. E. Goldberg. Analysis of mixing in genetic algorithms: A survey. Technical report, IlliGAL Report No. 2002012, Urbana, IL: University of Illinois at Urbana-Champaign, 2002.

K. Sastry and D. E. Goldberg. Probabilistic model building and competent genetic programming. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practice*, pages 205–220. Kluwer, 2003a.

K. Sastry and D. E. Goldberg. Scalability of selectorecombinative genetic algorithms for problems with tight linkage. In *GECCO'03: Proceedings of the 2003 Genetic and Evolutionary Computation Conference*, pages 1332–1344, 2003b.

K. Sastry and D. E. Goldberg. Designing competent mutation operators via probabilistic model building of neighborhoods. In *GECCO'04: Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, volume 2, pages 114–125, 2004.

K. Sastry and A. Orriols-Puig. Extended compact genetic algorithm in matlab. Technical report, IlliGAL Report No. 2007009, Urbana-Champaign IL 61801, USA, 2007.

H. P. Schwefel. *Numerical optimization of computer models.* John Wiley & Sons, 1981.

D. Sheskin. *Handbook of parametric and nonparametric statistical procedures.* Chapman & Hall, 2000.

R. E. Smith and M. Valenzuela-Rendón. A study of rule set development in learning classifier system. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 340–346, San Mateo, California, 1989. Morgan Kaufmann. ISBN 1-55860-006-3.

S. F. Smith. Flexible learning of problem solving heuristics through adaptive search. In *In Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 421–425, Los Altos, CA, 1983. Morgan Kaufmann.

S. F. Smith. Adaptive learning systems. In R. Forsyth, editor, *Expert Systems: Principles and Case Studies*, pages 168–189. Chapman and Hall, London, U.K., 1984.

S. F. Smith. *A learning system based on genetic adaptive algorithms.* PhD thesis, University of Pittsburgh, USA, 1980.

S. Y. Sohn. Meta analysis of classification algorithms for pattern recognition. *IEEE Transactions of Pattern Analyisis and Machine Learning*, 21(11):1137–1144, 1999.

C. Stone and L. Bull. For real! XCS with continuous-valued inputs. *Evolutionary Computation*, 11(3):299–336, 2003.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* Cambridge, MA: MIT Press, 1998.

K. Tamee, L. Bull, and O. Pinngern. A learning classifier system approach to clustering. In *ISDA'06: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications*, pages 621–626, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2528-8. doi: http://dx.doi.org/10.1109/ISDA.2006.62.

K. Tamee, L. Bull, and O. Pinngern. Towards clustering with XCS. In *GECCO'07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1854–1860, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4. doi: http://doi.acm.org/10.1145/1276958.1277326.

F. Teixidó-Navarro, A. Orriols-Puig, and E. Bernadó-Mansilla. Hierarchical evolution of linear regressors. In *GECCO'08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1413–1420, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-130-9. doi: http://doi.acm.org/10.1145/1389095.1389367.

S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Elsevier, 3rd edition, 2006.

D. Thierens and D. E. Goldberg. Convergence models of genetic algorithm selection schemes. In *Parallel Problem Solving from Nature*, volume 3, pages 116–121, 1994a.

D. Thierens and D. E. Goldberg. Elitist recombination: An integrated selection recombination ga. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 508–512, 1994b.

D. Thierens, D. E. Goldberg, and A. G. Pereira. Domino convergence, drift, and the temporal-salience structure of problems. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, page 535540, 1998.

P. Thrift. Fuzzy logic synthesis with genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the fourth International Conference on Genetic Algorithms*, pages 509–513. Morgan Kaufmann, 1991.

I. Tomek. Two modifications of CNN. *IEEE Transactions on Systems, Man and Cybernetics*, 6:769–772, 1976.

M. Valenzuela-Rendón. The fuzzy classifier system: A classifier system for continuously varying variables. In *4th ICGA*, pages 346–353. Morgan Kaufmann, 1991.

R. M. M. Vallim, D. E. Goldberg, X. Llorà, T. S. P. C. Duque, and A. C. P. L. F. Carvalho. A new approach for multi-label classification based on default hierarchies and organizational learning. In *GECCO'08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 2017–2022, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-131-6. doi: http://doi.acm.org/10.1145/1388969.1389015.

V. Vapnik. *The nature of statistical learning theory.* Springer Verlag, New York, 1995.

J. Velasco. Genetic-based on-line learning for fuzzy process control. *International Journal of Intelligent Systems*, 13:891–903, 1998.

G. Venturini. SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts. In P. B. Brazdil, editor, *Machine Learning: ECML-93 - Proc. of the European Conference on Machine Learning*, pages 280–296. Springer-Verlag, Berlin, Heidelberg, 1993.

G. Venturini. *Apprentissage adaptatif et apprentissage supervisé par algorithme génétique.* PhD thesis, Université de Paris-Sud, 1994.

C. Watkins. *Learning from delayed rewards.* PhD thesis, Cambridge University, 1989.

G. M. Weiss. Mining with rarity: a unifying framework. *SIGKDD Explorations Newsletter, special issue on learning from imbalanced datasets*, 6(1):7–19, 2004.

G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, editors, *Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 359–363, New York, NY, 1998. AAAI Press, Menlo Park, CA.

G. M. Weiss and F. Provost. Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.

B. Widrow and M. E. Hoff. Adaptive switching circuits. *Neurocomputing: foundations of research*, pages 123–134, 1988.

B. Widrow and M. A. Lehr. 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990. doi: 10.1109/5.58323.

D. Wierstra, F. Gómez, and J. Schmidhuber. Modeling systems with internal state using evolino. In *GECCO'05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1795–1802. ACM Press, 2005.

J. R. Wilcox. Organizational learning within a learning classifier system. Master's thesis, University of Illinois at Urbana Champaign, 1995.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

S. W. Wilson. Get real! XCS with continuous-valued inputs. In *Learning Classifier Systems. From Foundations to Applications*, LNAI, pages 209–219, Berlin, 2000. Springer-Verlag.

S. W. Wilson. Mining oblique data with XCS. In *IWLCS'00: Revised Papers from the Third International Workshop on Advances in Learning Classifier Systems*, pages 158–176, London, UK, 2001. Springer-Verlag. ISBN 3-540-42437-7.

S. W. Wilson. Compact rulesets from XCSI. In P. L. Lanzi, W. Stolzmann, and S. W. Wilson, editors, *Advances in Learning Classifier Systems, 4th International Workshop*, volume 2321 of *Lecture Notes in Artificial Intelligence*, pages 197–210. Springer, 2002a. ISBN 3-540-43793-2.

S. W. Wilson. Classifiers that approximate functions. *Journal of Natural Computing*, 1(2): 211–234, 2002b.

S. W. Wilson. Classifier conditions using gene expression programming. Technical report, IlliGAL Report No. 2008001, Urbana-Champaign IL 61801, USA, 2008.

S. W. Wilson. Aubert processing and intelligent vision. Technical report, Polaroid Corporation, Cambridge, MA, 1981.

S. W. Wilson. Adaptive "cortical" pattern recognition. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 188–196, 1985a.

S. W. Wilson. Knowledge growth in an artificial animal. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 16–23, Mahwah, NJ, USA, 1985b. Lawrence Erlbaum Associates, Inc. ISBN 0-8058-0426-9.

S. W. Wilson. Classifier systems and the animat problem. *Maching Learning*, 2(3):199–228, 1987. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1022655214215.

S. W. Wilson. ZCS: a zeroth level classifier system. *Evolutionary Computation*, pages 1–18, 1994.

S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

S. W. Wilson. Generalization in the XCS classifier system. In *3rd Annual Conf. on Genetic Programming*, pages 665–674. Morgan Kaufmann, 1998.

S. W. Wilson and D. E. Goldberg. A critical review of classifier systems. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 244–255, 1989.

I. Witten and E. Frank. *Data mining: practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2nd edition, 2005. ISBN 0-12-088407-0.

D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2007. doi: 10.1007/s10115-007-0114-2.

I. Yalabik and T. Y.-V. Fatos. A pattern classification approach for boosting with genetic algorithms. *22nd International International Symposium on Computer and Information Sciences, 2007. ISCIS 2007*, pages 1–6, 2007. doi: 10.1109/ISCIS.2007.4456870.

L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:28–44, 1973.

Index